1 Topic

Studying the influence of two approaches for the treatment of missing values (univariate imputation and listwise deletion) on the performances of the logistic regression.

The handling of missing data is a difficult problem. Not because of its management which is simple, we just report the missing value with a specific code, but rather because of the consequences of their treatment on the characteristics of the models learned on the treated data.

We have already analyzed this problem in a previous paper. We studied the impact of the different techniques of missing values treatment on a decision tree learning algorithm (C4.5; Quinlan 1993)[1]. In this paper, we repeat the analysis by examining their influence on the results of the logistic regression. In a first time, we mainly use the **R** software, with the **glm()** procedure. In a second time, we examine the behavior of the tools proposed by **Orange**, **Knime** and **RapidMiner**.

The origin of the missing value[2] is the first characteristic which determines their treatment. It can be **MCAR** (missing completely at random) i.e. the apparition of the missing value does not depend neither to the values of the variable, nor to the values of the other variables of the database. It can be **MAR** (missing at random) i.e. its occurrence does not depend on the values of the variable after we remove the influence of the other variables. For instance, managers are less likely to declare their salaries in a survey. But in this category, the probability of the occurrence of missing value does not depend on the value of the salary. Last, it can be **MNAR** (missing not at random) i.e. the occurrence of the missing value is related to the true value of the variable. We cannot use a statistical approach for the imputation in this case.

Among these configurations, the MCAR case is the easier to manage. We get unbiased estimates of parameters, even when we use very simplistic techniques such as the listwise deletion approach[3].

The characteristic of the subsequent statistical method applied on the pretreated dataset is the second important piece of the puzzle. Clearly, the influence of missing values and their treatment is not the same on a predictive analysis (e.g. logistic regression), on a clustering algorithm (e.g. an Agglomerative Hierarchical Clustering) or on a factor analysis (e.g. a principal component analysis). The distinction is all the more important that some statistical methods incorporate natively a strategy for the handling of missing values (e.g. the NIPALS algorithm[4] for principal components analysis and the PLS regression).

Last, the criteria for assessing the treatment of missing data is the final piece of the puzzle. The goal is not to find the "true" values of missing observations for each variable. But rather to propose replacement values (in the case of imputation) which do not alter the results of the study we are conducting. We highlight often bias and the variance of the estimated parameters in the statistical

[1] http://data-mining-tutorials.blogspot.fr/2009/11/handling-missing-values-in-sipina.html

[2] http://www.uvm.edu/~dhowell/StatPages/More_Stuff/Missing_Data/Missing.html

[3] http://en.wikipedia.org/wiki/Listwise_deletion

[4] http://en.wikipedia.org/wiki/Non-linear_iterative_partial_least_squares

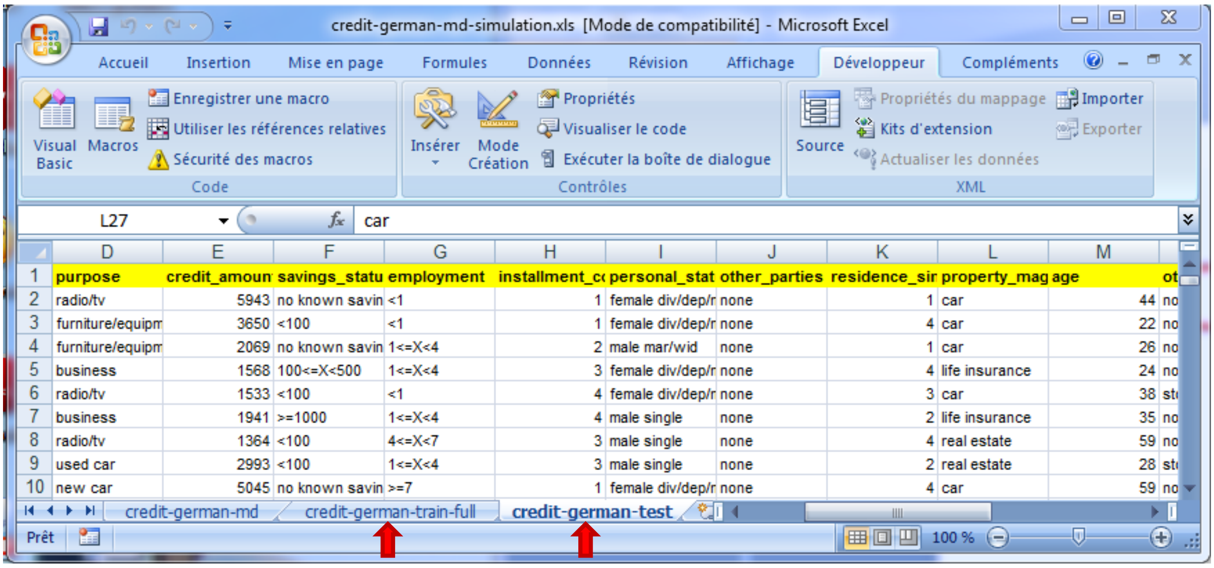
literature. But in the specific context of predictive analysis, we can also ask the question of performance. What is the missing value processing technique which enables to build the most efficient model in prediction?

In this tutorial, we consider the following configuration: (1) missing values are MCAR, we wrote a program which removes randomly some values in the learning sample; (2) we apply logistic regression on the pre-treated training data i.e. on a dataset on which we apply a missing value processing technique; (3) we evaluate the different techniques of treatment of missing data by observing the accuracy rate of the classifier on a separate test sample which has no missing values.

In a first time, we conduct the experiments with R. We compare the listwise deletion approach to the univariate imputation (the mean for the quantitative variables, the mode for the categorical ones). We will see that this latter is a very viable approach in MCAR situation. In a second time, we will study the available tools in Orange, Knime and RapidMiner. We will observe that despite their sophistication, they are not better than the univariate imputation in our context.

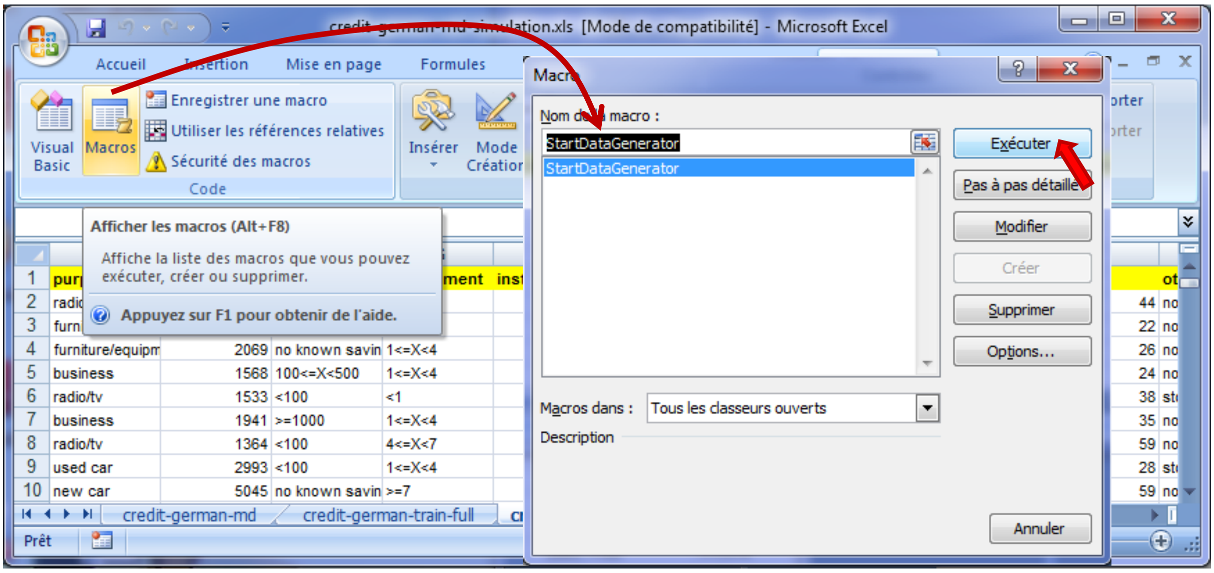
2 Dataset

Original dataset. We use the GERMAN CREDIT DATA in our experiment[5]. We randomly split the data file into two sheets in an Excel workbook (**credit-german-md-simulation.xls**): CREDIT-GERMAN-TRAIN-FULL (300 instances) is used for the learning model phase; CREDIT-GERMAN-TEST (700 instances) for the test phase. For this second sample, we created once and for all the data file CREDIT-GERMAN-TEST.TXT (text file format). There are no missing values in these two samples.

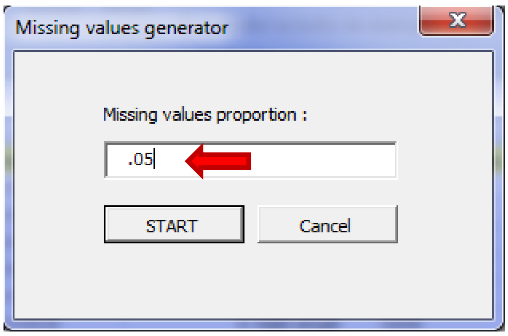


Creating a training sample with missing values. We use a VBA program to generate a learning sample with some missing values. For this, we launch the program by clicking on the DEVELOPPEUR / MACROS into Excel (*for the French version of Excel*). Into the dialog box which appears, we select the STARTDATAGENERATOR program.

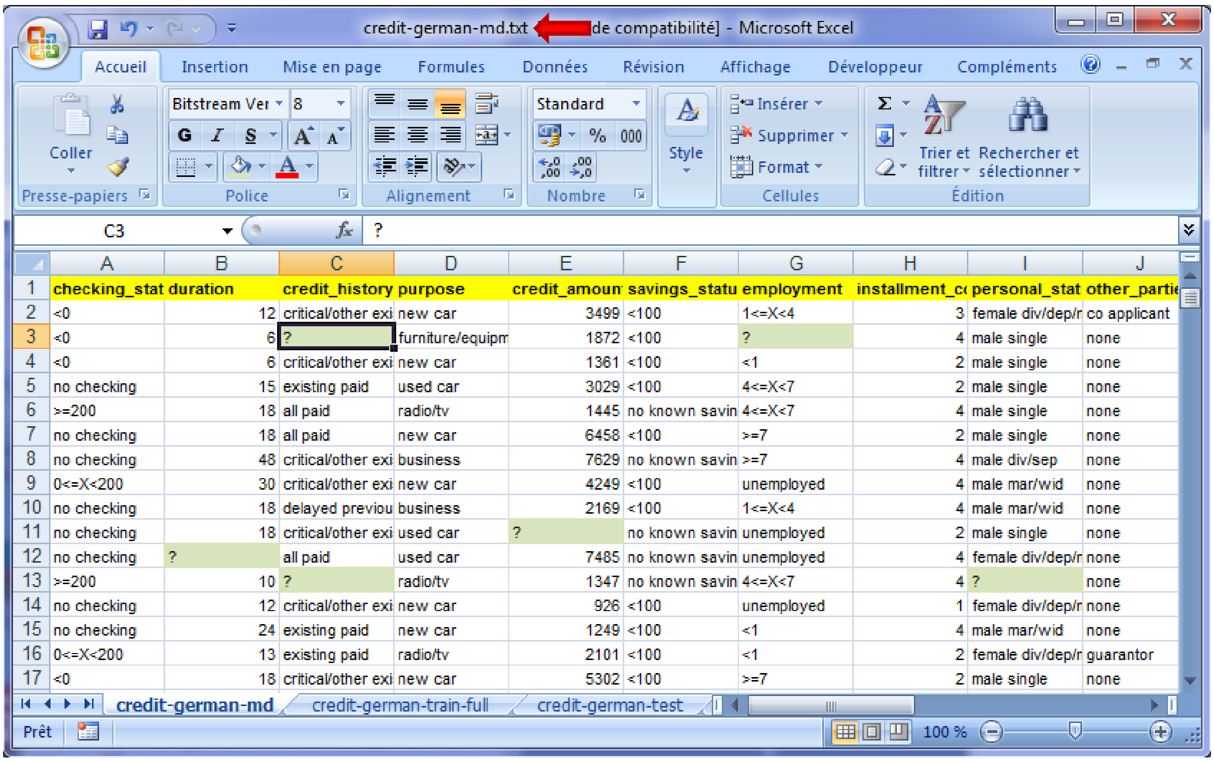
[5] http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29



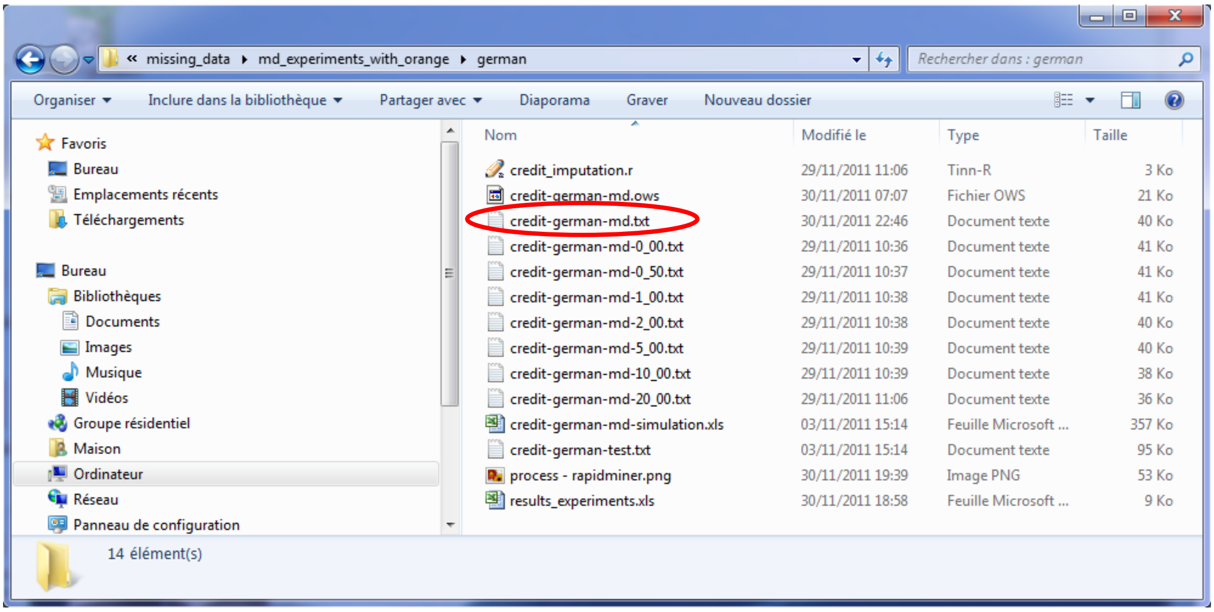
The program is started, a settings dialog asks the proportion of the missing values that we want to insert into the learning sample. For instance, if we set 0.05 (5%), the program generates a learning sample with (0.05 * 300 rows * 20 variables = 300 cells) missing values (the '?' character is used for materializing the missing value). The class attribute is not concerned by this process.



The resulting dataset is inserted in the first sheet of the workbook (CREDIT-GERMAN-MD).



At the same time, the learning sample 'CREDIT-GERMAN-MD.TXT' (text file format) is automatically created in our working directory (the directory of the XLS file).



We use the following VBA code for this task.

```
Private Sub cmdBOk_Click()
'retrieve the proportion from a dialogbox
Dim proportion As Double
proportion = Val(tbProportion.Text)
'checking the range of the value (< 0.5 max.)
If (proportion < 0#) Or (proportion >= 0.5) Then
    proportion = 0.01
End If
'copying the full training set to the first sheet
Sheets("credit-german-train-full").Select
Range("A2:U301").Select
Selection.Copy
Sheets("credit-german-md").Select
Range("A2").Select
ActiveSheet.Paste
'number of cells to clear
Dim nbMd As Long
nbMd = Int(6000# * proportion)
'counter for the number of missing values
Dim counter As Long
counter = 1
'coordinates
Dim i As Long, j As Long
'initialize the random number generator
Randomize (100)
Do While (counter <= nbMd)
    'row
    i = 2 + Int(300# * Rnd)
```

```
    'column
    j = 1 + Int(20# * Rnd)
    'checking if the cell is already empty
    If (Cells(i, j).Value <> "?") Then
        Cells(i, j).Value = "?"
        counter = counter + 1
    End If
Loop
'set the text file name
Dim nomFichier As String
nomFichier = "\credit-german-md.txt"
nomFichier = ThisWorkbook.Path + nomFichier
'checking if the file already exists
Dim fs As Variant
Set fs = CreateObject("Scripting.FileSystemObject")
If (fs.FileExists(nomFichier) = True) Then
    fs.deletefile (nomFichier)
End If
'save the sheet into a text file
ThisWorkbook.SaveAs fileName:=nomFichier, FileFormat:=xlTextWindows
'close the form (dialogbox for the proportion setting)
formMD.Hide
End Sub
```

We note that if we insert 300 missing value in our dataset, it does not mean that we have 300 observations with at least one missing value. Some instances have more than on missing value (e.g the row #13 into the screenshot above), others are complete. In the follows, we count the number of instances with no missing values in each generated learning samples.

3 Processing the missing values with R

3.1 The R program

We wish to compare two methods of dealing with missing data: the listwise deletion and univariate imputation (replacement by the average for the quantitative variables, by the mode for the categorical ones). To evaluate the efficiency of the strategies, we apply the models learned from the pre-treated dataset on the same test sample with no missing values. The best strategy is the one that induces the best model i.e. which has the best test accuracy rate.

The R program is subdivided in several parts.

Loading the samples. We load the learning sample (with some missing values) and the test sample (with no missing values).

```
#loading the test sample
setwd("… votre répertoire …")
data.test <- read.table(file="credit-german-test.txt",dec=".",sep="\t",header=T)
#loading the test sample
```

```
data.train <- read.table(file="credit-german-md.txt",dec=".",sep="\t",header=T,na.strings="?")
summary(data.train)
```

Listwise deletion approach. The first strategy is to remove from the dataset the observations which contain at least one missing value. This is the listwise deletion strategy. We can reduce dramatically the dataset size with this approach. But in the MCAR context, with a moderate proportion of missing values, it can be acceptable.

```
#first approach: listwise deletion
data.train.omitted <- na.omit(data.train)
summary(data.train.omitted)
print(paste('Remaining observations : ',as.character(nrow(data.train.omitted))))
model.omitted <- glm(classe ~ ., family = binomial, data = data.train.omitted)
```

na.omit(.) removes from the dataset the instances with at least one missing value. **model.omitted** is the model learned from the resulting observations.

Univariate imputation. We use the following program for the univariate imputation. We obtain the **model.imputed** model from the pre-treated learning sample.

```
#second approach: univariate imputation univariée (mean, mode)
data.train.imputed <- as.data.frame(lapply(data.train,traiter_missing_data))
summary(data.train.imputed)
model.imputed <- glm(classe ~ ., family = binomial, data = data.train.imputed)
```

The **traiter_missing_data(x)** function makes the distinction between the continuous (numeric) and the factor (categorical) variables.

```
#according to the variable type
traiter_missing_data <- function(x){
 if (is.factor(x) == T){
  return(traiter.discrete(x))
 } else {
  return(traiter.numeric(x))
 }
}
```

Thus, for the continuous attribute, we calculate the mean on the available values and we replace the NA by this value:

```
#continuous variable, replace NA by the mean
traiter.numeric <- function(x){
 y <- x
 z <- na.omit(y)
 if (length(z) < length(y)){
  m <- mean(z)
  y[is.na(y)] <- m
```

```
 }
 return(y)
}
```

The approach is similar for the categorical variables; but we use the 'mode'[6] instead the mean:

```
#categorical variable, replace NA by the mode
traiter.discrete <- function(x){
 y <- x
 z <- na.omit(y)
 if (length(z) < length(y)){
  frequence <- table(z)
  m <- which.max(frequence)
  y[is.na(y)] <- levels(x)[m]
 }
 return(y)
}
```

Model evaluation. Last step of our experiment, we evaluate the models. We use a specific function for this. It takes as input the test sample and the model. It computes the model prediction, the confusion matrix and the accuracy rate. We use the accuracy rate instead of the error rate to obtain results directly comparable to the outputs of the other tools.

```
#new.dataset is the test sample
#model is the model to evaluate
pred_and_confusion_matrix <- function(new.dataset,model){
 #score predicted by the model
 data.pred.prob <- predict(model,newdata = new.dataset)
 #classification from the score
 data.pred.class <- ifelse(data.pred.prob > 0.5,"B","A")
 data.pred.class <- as.factor(data.pred.class)
 #confusion matrix (observed class values vs. predicted value)
 mc <- table(new.dataset$classe,data.pred.class)
 print(mc)
 #accuracy rate
 ca <- (mc[1,1]+mc[2,2])/sum(mc)
 print(ca)
}
```

3.2 Organization of the experiments

The idea is to observe the impact of the treatment strategy when the proportion of missing values increases. We try the following percentages: 0.5%, 1%, 2%, 5%, 10% and 20%.

---

[6] http://en.wikipedia.org/wiki/Mode_%28statistics%29

The reference result is the performance of the model learned from the complete training sample (0% missing data). Of course, both strategies should provide the same result since no missing value processing is performed. The test accuracy rate of the model is **73%**. We should not obtain a better result for the model learned on the learning samples with missing values below (normally).

```
> #première solution : listwise deletion
> data.train.omitted <- na.omit(data.train)
> #summary(data.train.omitted)
> print(paste('Remaining observations : ',as.character(nrow(data.train.omitted))))
[1] "Remaining observations :  300"
> model.omitted <- glm(classe ~ ., family = binomial, data = data.train.omitted)
> #évaluation
> pred_and_confusion_matrix(data.test,model.omitted)
      data.pred.class
         A   B
  bad  107 103
  good  86 404
[1] 0.73
>
> #seconde solution : imputation univariée (moyenne, mode)
> data.train.imputed <- as.data.frame(lapply(data.train,traiter_missing_data))
> #summary(data.train.imputed)
> model.imputed <- glm(classe ~ ., family = binomial, data = data.train.imputed)
> #évaluation
> pred_and_confusion_matrix(data.test,model.imputed)
      data.pred.class
         A   B
  bad  107 103
  good  86 404
[1] 0.73
>
> |
```

3.3 Experiment results

We obtain the results into the table below.

GERMAN DATASET		Accuracy rate	
% missing	# complete obs.	Listwise Del.	Univ. Imputation
0,00%	300	0,7300	0,7300
0,50%	272	0,7100	0,7257
1,00%	246	0,7129	0,7286
2,00%	201	0,7214	0,7186
5,00%	111	0,6729	0,7114
10,00%	40	ERR	0,7086
20,00%	4	ERR	0,7214

We observe that:

- The presence of the missing values deteriorates the model performance.
- When there is a little proportion of missing values (up to 2%, about 201 complete instances), the two approaches give similar performances.

- When the proportion is higher, the listwise deletion approach seems inappropriate. Starting from a certain proportion (10%), the learning process is impossible because there are too few complete instances into the dataset.
- In contrast, the univariate imputation has a really good behavior. In the configuration MCAR, the strategy is appropriate even when there is 20% missing values in the training sample. In fact, this is not surprising. The "?" are well spread throughout the columns, there is finally a few proportion of missing observations for each variable. The value used for the replacement is viable. For the proportion of 20%, about 240 values are available in each column.

Again, we are in the perfect MCAR context in this tutorial. We must not forget it when we read the results. However, they are really interesting. Particularly, the univariate imputation seems viable in this context, even when the proportion of missing values is high.

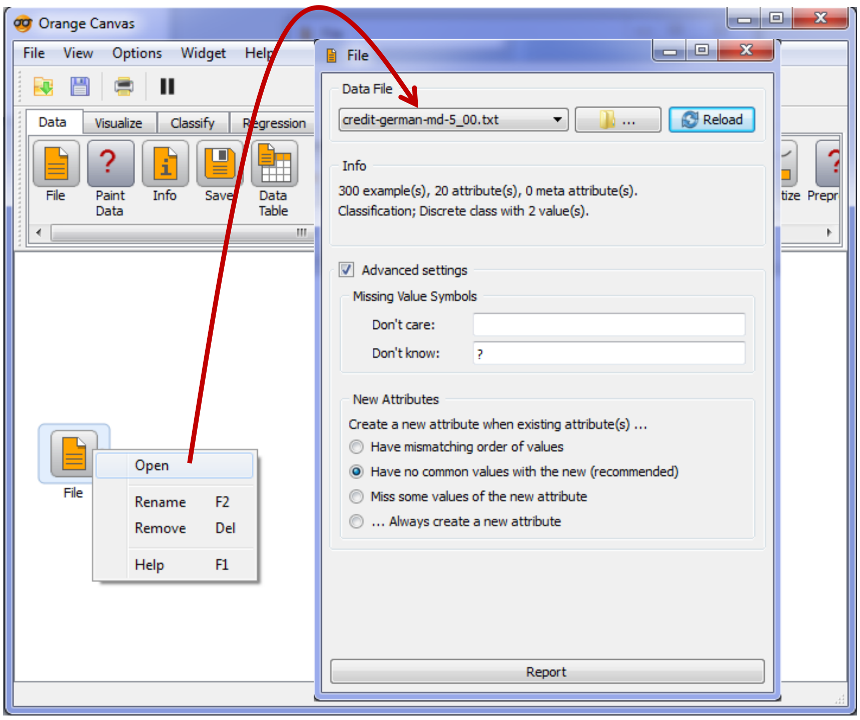
4 Processing missing values in other software

In this section, we describe the tools for processing missing values in various data mining software. We work on the learning sample with 5% of missing values (111 complete instances on 300). The test sample is the same as previously.

4.1 ORANGE

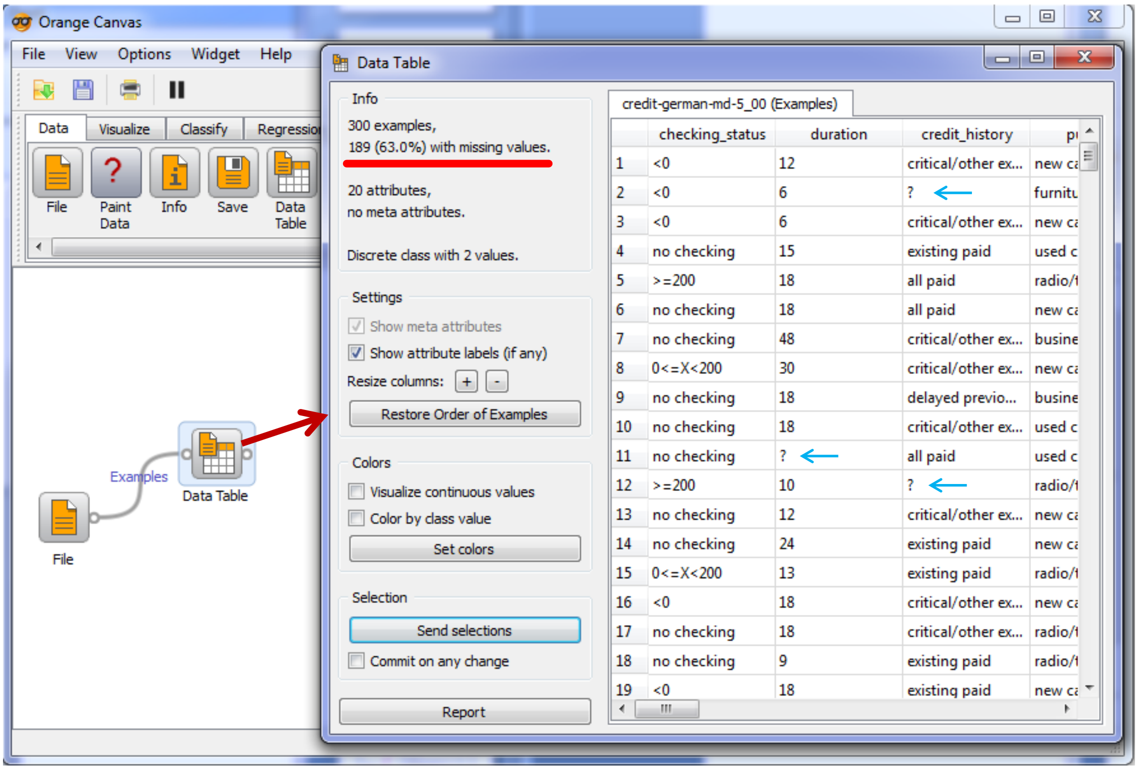
4.1.1 Programming the analysis

Missing value imputation. After we launch Orange, we can define a new schema.



We set the FILE (DATA tab) component into the canvas. We set the file name (CREDIT-GERMAN-MD-5_00.TXT) and we specify the code '?' (DON'T KNOW) which represents the missing value.

We visualize the dataset with the DATA TABLE (DATA tab) tool.



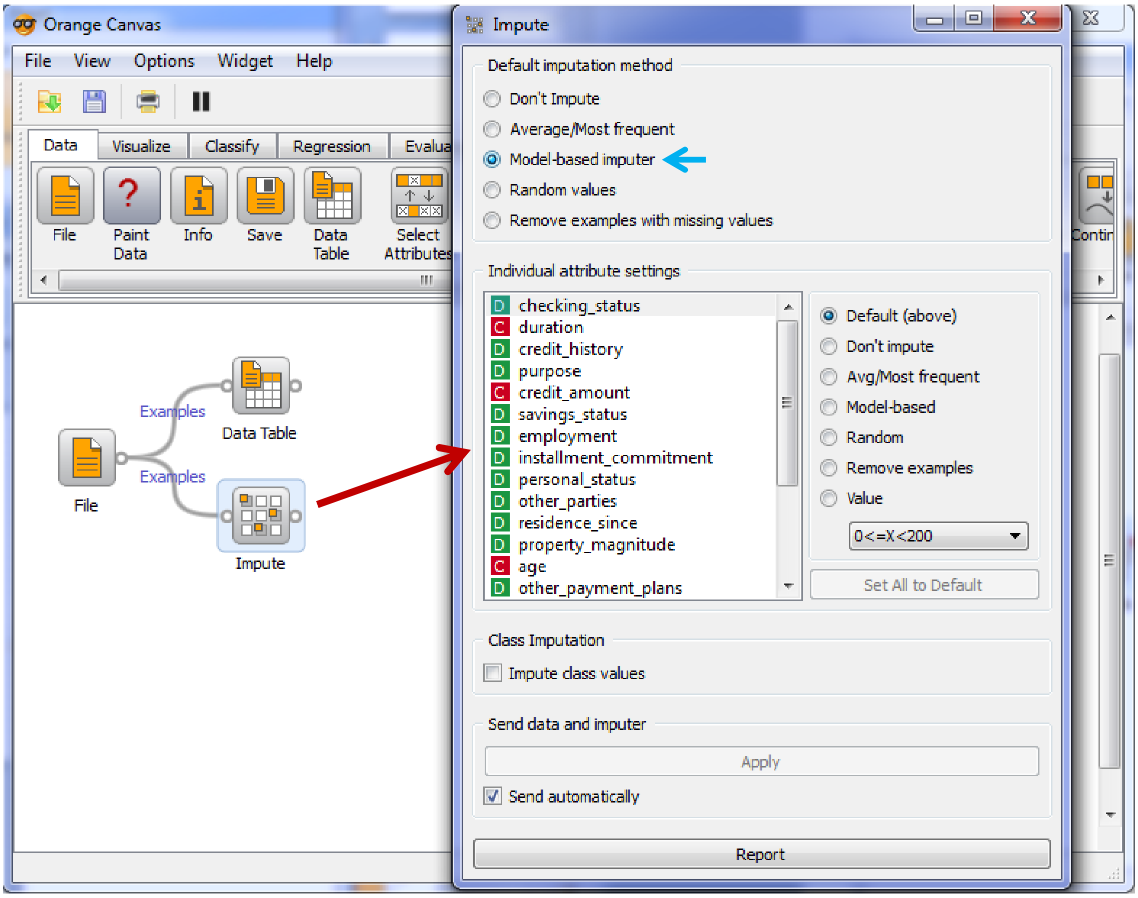
Orange announced that 189 observations contain at least one missing value (189 + 111 complete instances = 300). It handles properly the missing value code during the importation.

We insert the IMPUTE (DATA tab) tool into the canvas. We click on the OPEN menu to set the parameters. The following options are available:

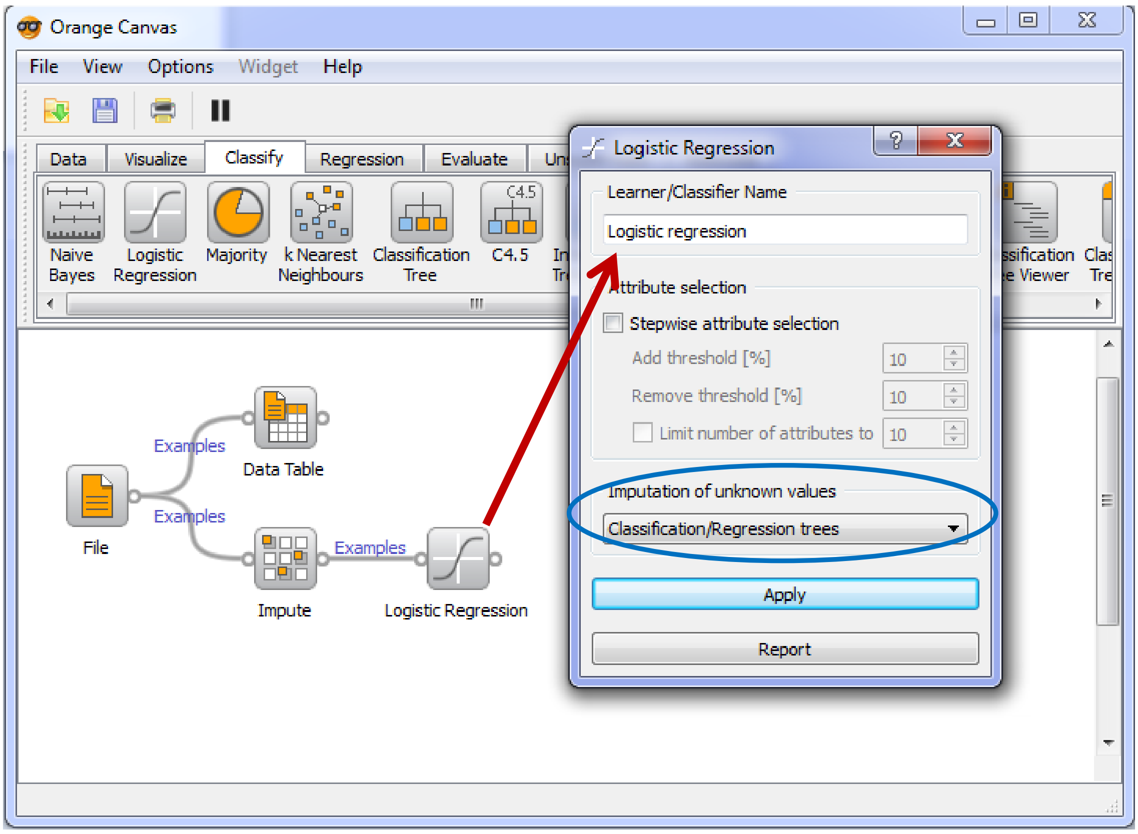
- DON'T IMPUTE. Nothing is done. The subsequent learning algorithm (e.g. LOGISTIC REGRESSION) handles the missing values.
- AVERAGE /MOST FREQUENT. This is the **univariate imputation** described above.
- MODEL-BASED IMPUTER. It uses a nearest neighbor algorithm to replace the value of a variable from the values of the other variables. This approach seems better than the univariate one because we make use of the relation between the variables. But the calculation time may be prohibitive on a large dataset. This approach is known also as the hot deck imputation strategy.
- RANDOM VALUES. Orange tries to approximate the distribution of each variable. It then uses a value retrieved in accordance with approximated distribution into the range of possible values.
- REMOVE EXAMPLES WITH MISSING VALUES. This is the "**listwise deletion**" strategy.

We choose the MODEL BASED IMPUTER approach.

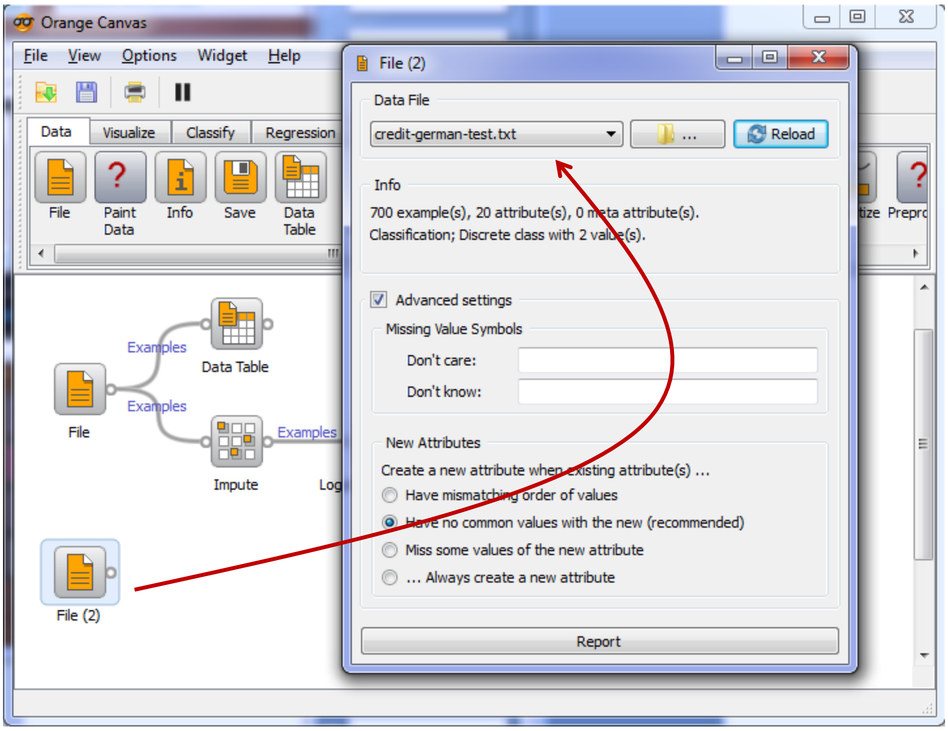
N.B.: We note that we can define a specific strategy for each variable of the database. We can even directly specify a value to replace the missing ones for each variable.



We add the LOGISTIC REGRESSION tool (CLASSIFY tab). We do not use the variable selection process.

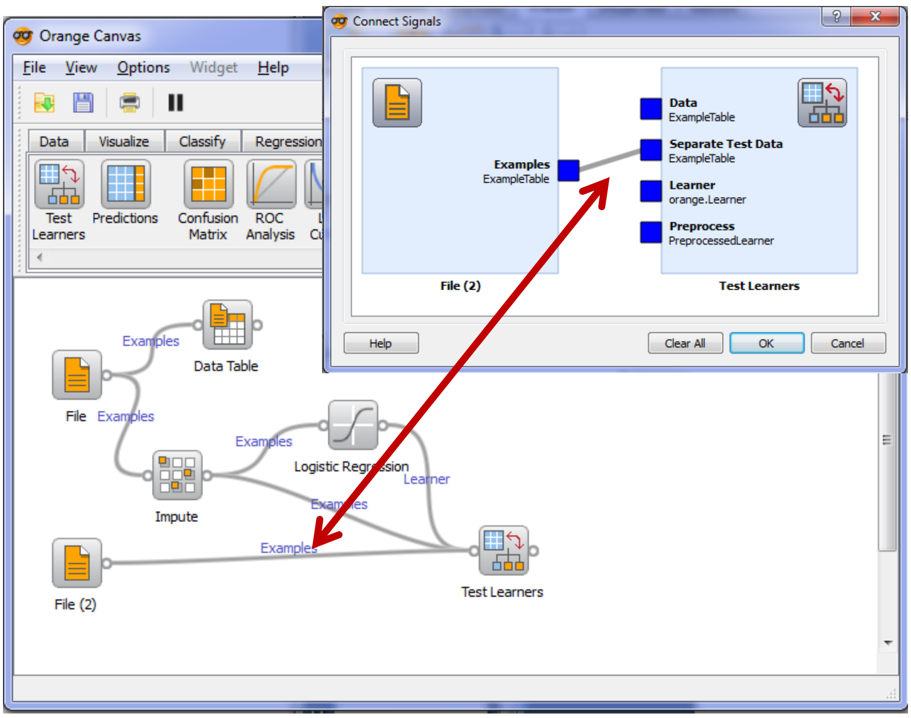


We note that the component "logistic regression" has an internal mechanism for the imputation of missing values (IMPUTATION OF UNKNOWN VALUES). Various solutions are proposed: univariate imputation (average, minimum, maximum); multivariate imputation where it predicts missing values of a predictive variable from the other variables in the database (using a decision/regression tree). This feature is enabled if we choose the DON'T IMPUTE option into the IMPUTE component.



Model evaluation on the test sample. The model is automatically learned when we close the setting dialog (we can visualize it using the NOMOGRAM tool[7]). We want to evaluate its performance on the test set. We insert again the FILE component and we select the CREDIT-GERMAN-TEST.TXT data file.

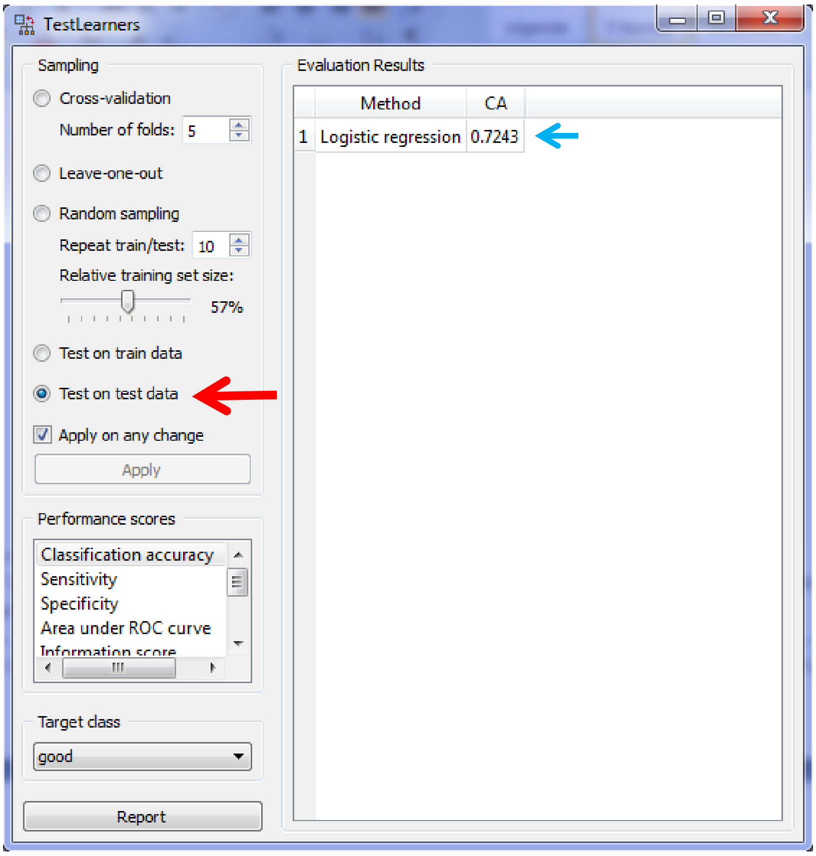
Then we add the TEST LEARNER component (EVALUATE tab). We set the following connections.



[7] E.g. http://data-mining-tutorials.blogspot.fr/2010/07/naive-bayes-classifier-for-discrete.html

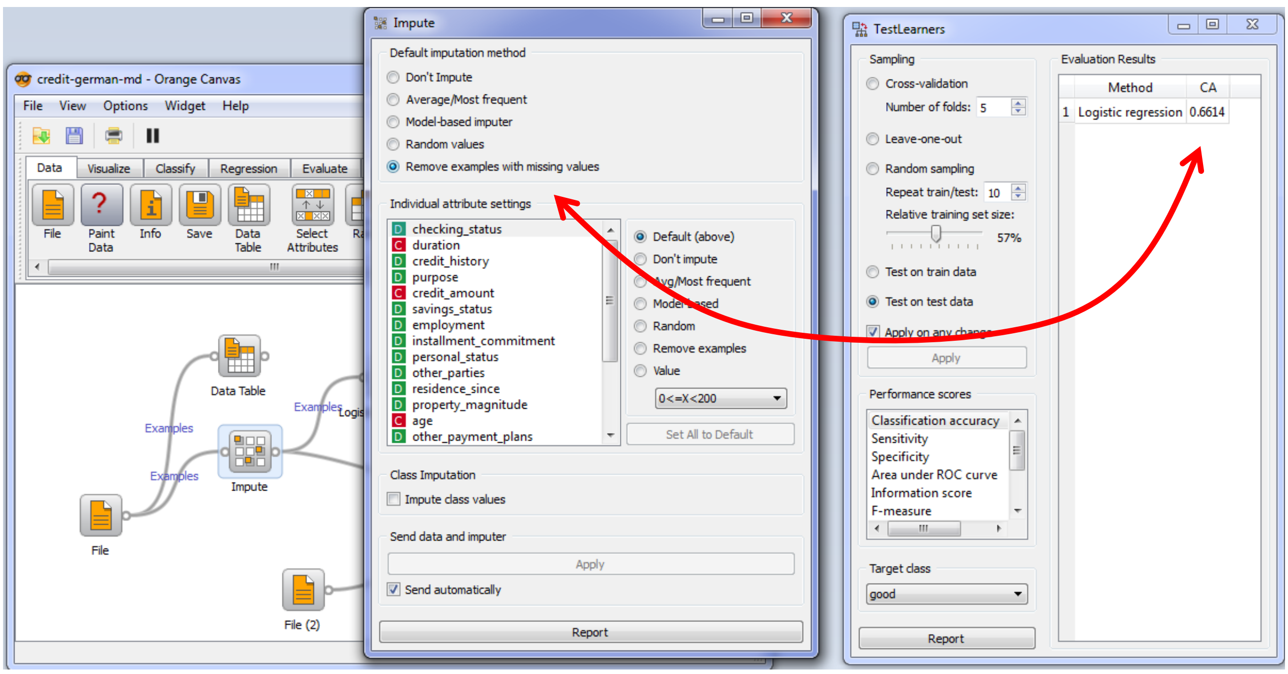
For the connection between the sample **FILE (2)** and the **TEST LEARNERS** component, we must specify that it concerns a separate test set (SEPARATE TEST DATA).

We open the TEST LEARNERS component. We specify that we evaluate the classifier on the test sample. The test accuracy rate is 72.43%.



4.1.2 Comparison of the imputation techniques

We want to compare the various imputation methods on our dataset. This is easy under Orange.



Orange proposes a very practical feature. We leave open the TEST LEARNERS window and we interactively modify the imputation option into IMPUTE component. The results are refreshed when we modify the option (SEND AUTOMATICALLY must be selected). We obtain the following table:

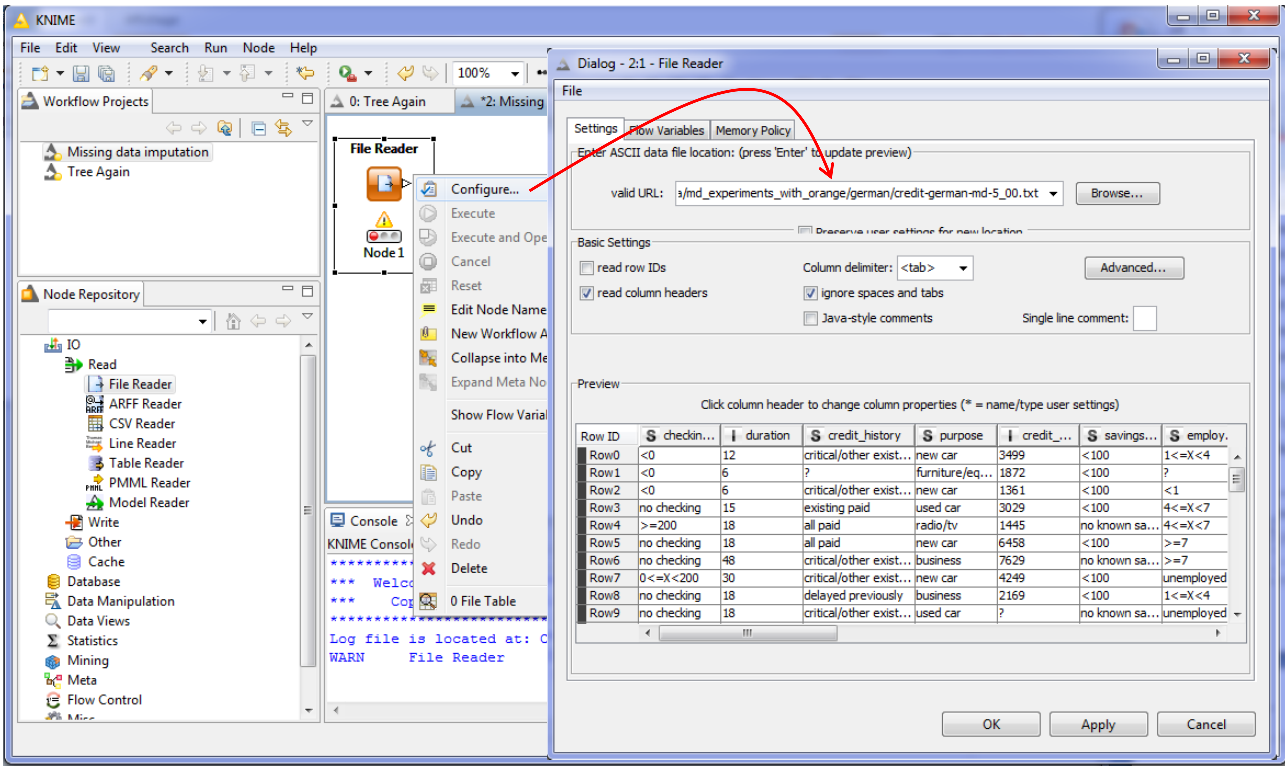
Approach	Class. Accuracy
Don't impute (Internal method of Logistic Reg.)	0.7357
Average / Most Frequent	0.7271
Model-based imputer	0.7243
Random Values	0.7357
Remove Examples with missing values (listwise deletion)	0.6614

Apart from listwise deletion approach, all methods are similar on our dataset. Curiously, the accuracy rate in some situations is higher than the accuracy of the learned model on the complete dataset under R (73%). Perhaps R and Orange[8] logistic regression algorithms have not exactly the same characteristics. It is possible also that it was simply the consequence of fluctuations sampling. Anyway, we find that the listwise deletion approach is not appropriate when the proportion of missing values increases (5% for the experiments under Orange).

4.2 KNIME

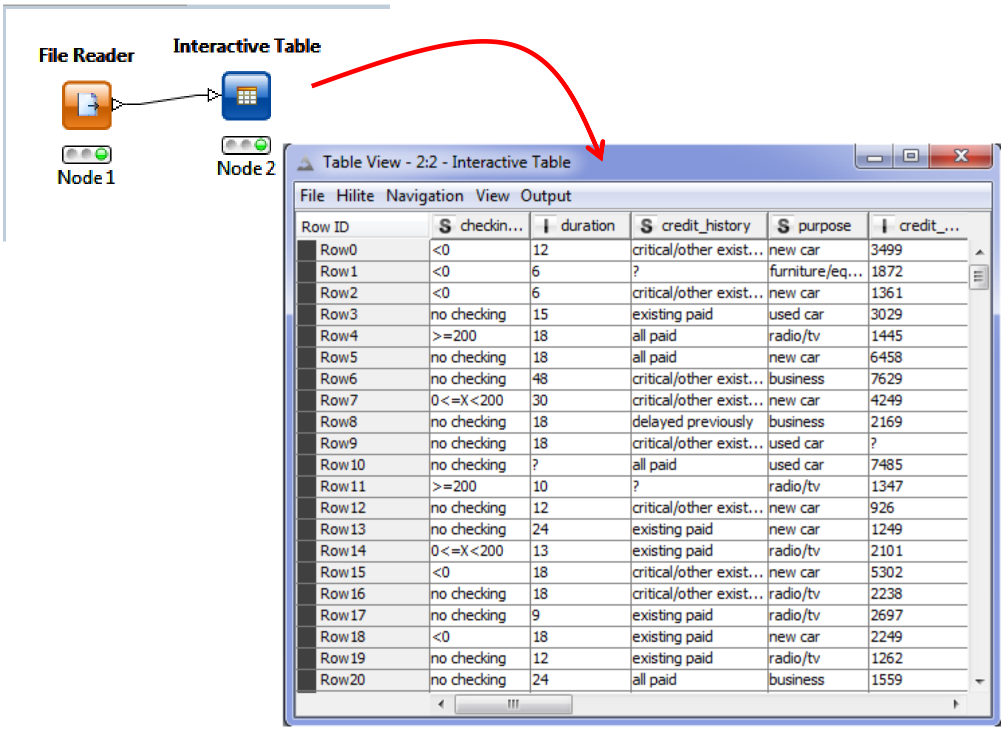
KNIME proposes the MISSING VALUES component for the treatment of missing data. In this section, we reproduce inthis section the analysis schema that we have defined under Orange.

Listwise deletion strategy. We import the CREDIT-GERMAN-MD-5_00.TXT data file using the FILE READER tool (IO / READ). The missing value is symbolized by the character '?'.



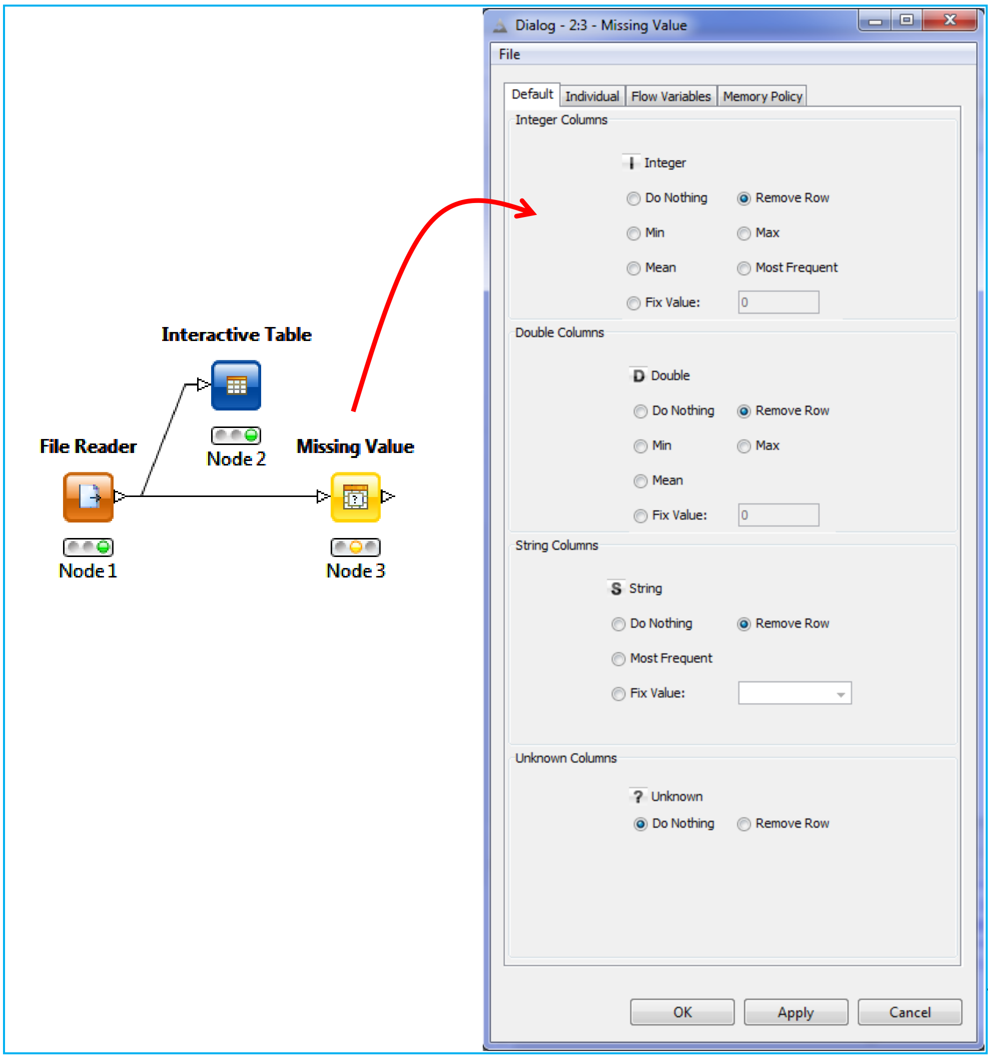
[8] http://orange.biolab.si/doc/reference/Orange.classification.logreg/

We launch the importation by clicking on the EXECUTE menu.

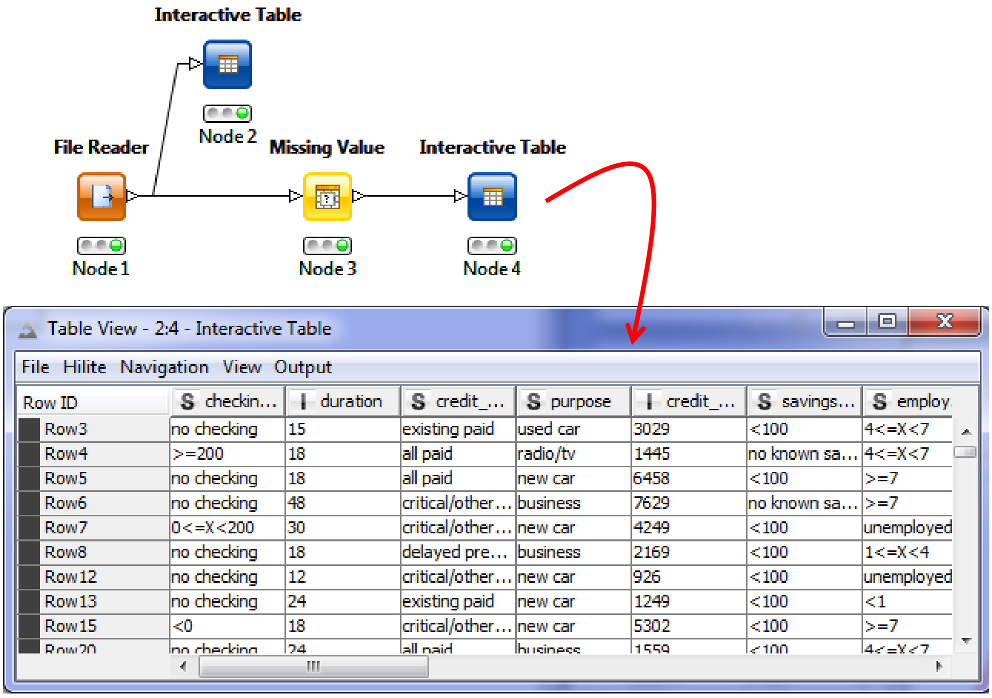


Row ID	S checkin...	I duration	S credit_history	S purpose	I credit_...
Row0	<0	12	critical/other exist...	new car	3499
Row1	<0	6	?	furniture/eq...	1872
Row2	<0	6	critical/other exist...	new car	1361
Row3	no checking	15	existing paid	used car	3029
Row4	>=200	18	all paid	radio/tv	1445
Row5	no checking	18	all paid	new car	6458
Row6	no checking	48	critical/other exist...	business	7629
Row7	0<=X<200	30	critical/other exist...	new car	4249
Row8	no checking	18	delayed previously	business	2169
Row9	no checking	18	critical/other exist...	used car	?
Row10	no checking	?	all paid	used car	7485
Row11	>=200	10	?	radio/tv	1347
Row12	no checking	12	critical/other exist...	new car	926
Row13	no checking	24	existing paid	new car	1249
Row14	0<=X<200	13	existing paid	radio/tv	2101
Row15	<0	18	critical/other exist...	new car	5302
Row16	no checking	18	critical/other exist...	radio/tv	2238
Row17	no checking	9	existing paid	radio/tv	2697
Row18	<0	18	existing paid	new car	2249
Row19	no checking	12	existing paid	radio/tv	1262
Row20	no checking	24	all paid	business	1559

To check the data importation, we visualize the dataset using the INTERACTIVE TABLE (DATA VIEWS) component

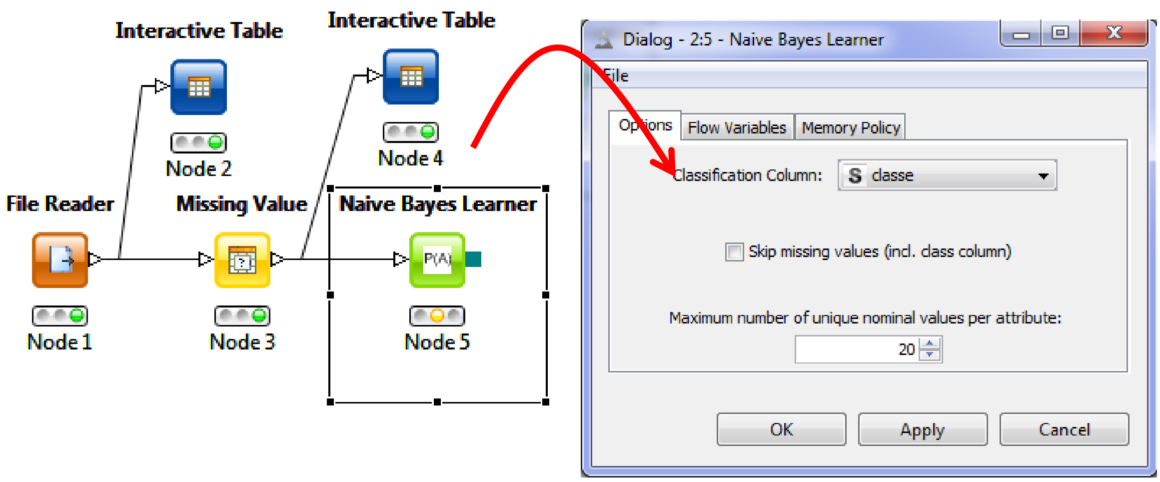


We insert now the component MISSING VALUE (DATA MANIPULATION / COLUMN / TRANSFORM) into the workflow. We implement the listwise deletion strategy (REMOVE ROW), whatever the variable type, into the settings dialog (CONFIGURE menu). Again, to check the result of the treatment, we visualize the dataset with the interactive table component. We observe that the rows containing at least one missing value (ROW0, ROW1, ROW2, ROW9, ROW10, etc.) are removed from the learning set.

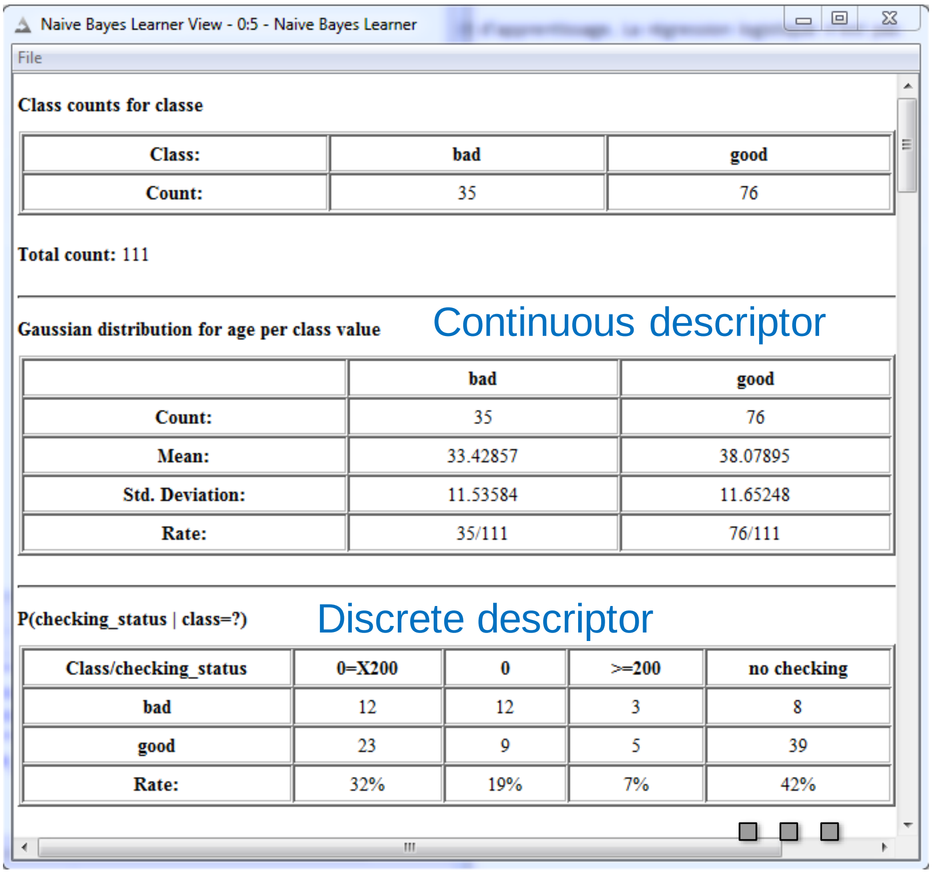


Naïve bayes classifier. In the first version - in French - of this tutorial, I did not see that the Logistic Regression is into the STATISTIC branch and not into the MINING branch into the node repository. I have thought that this approach was not available. I decided to use the naive bayes classifier, which is also a linear classifier. Subsequently, Loïc Lucel (thank you very much Loïc) had pointed out to me the presence of logistic regression. So I decided to keep in this tutorial the studies for the naive bayes classifier and the logistic regression.

We add the NAÏVE BAYES LEARNER (MINING / BAYES) into the workflow. We set CLASSE as classification column.



We click on the EXECUTE AND OPEN VIEWS menu to obtain the results. Knime provides the conditional mean and standard deviation for continuous descriptors, the contingency table for the calculation of the conditional probabilities for the discrete ones.



Naive Bayes Learner View - 0:5 - Naive Bayes Learner

File

Class counts for classe

Class:	bad	good
Count:	35	76

Total count: 111

Gaussian distribution for age per class value

Continuous descriptor

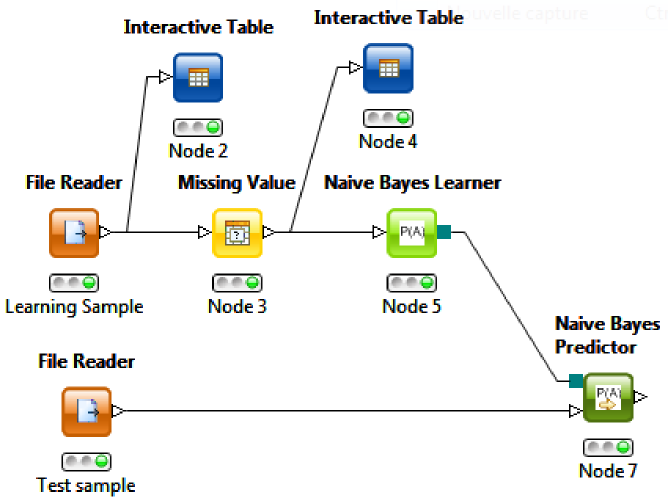
	bad	good
Count:	35	76
Mean:	33.42857	38.07895
Std. Deviation:	11.53584	11.65248
Rate:	35/111	76/111

P(checking_status | class=?)

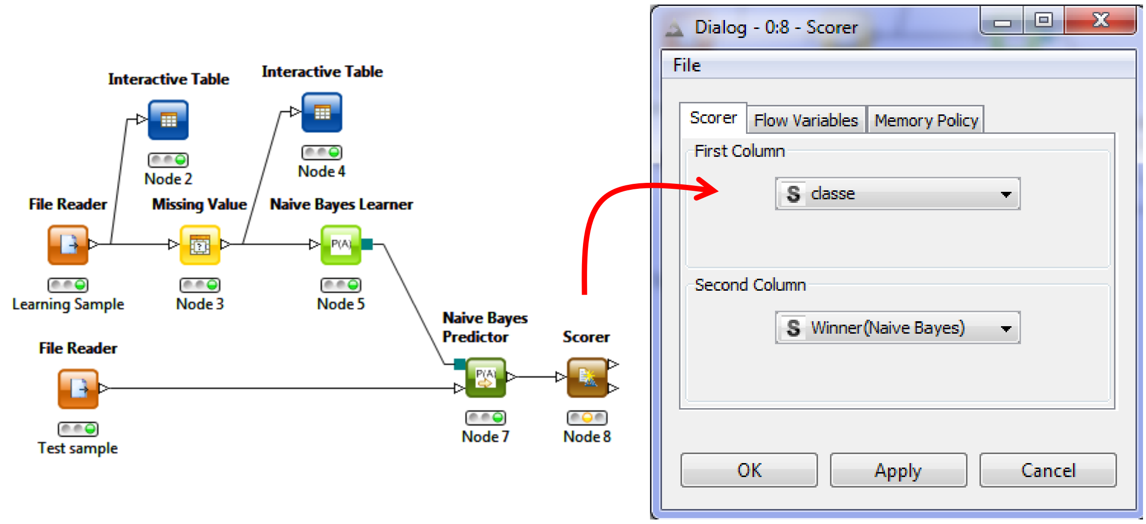
Discrete descriptor

Class/checking_status	0=X200	0	>=200	no checking
bad	12	12	3	8
good	23	9	5	39
Rate:	32%	19%	7%	42%

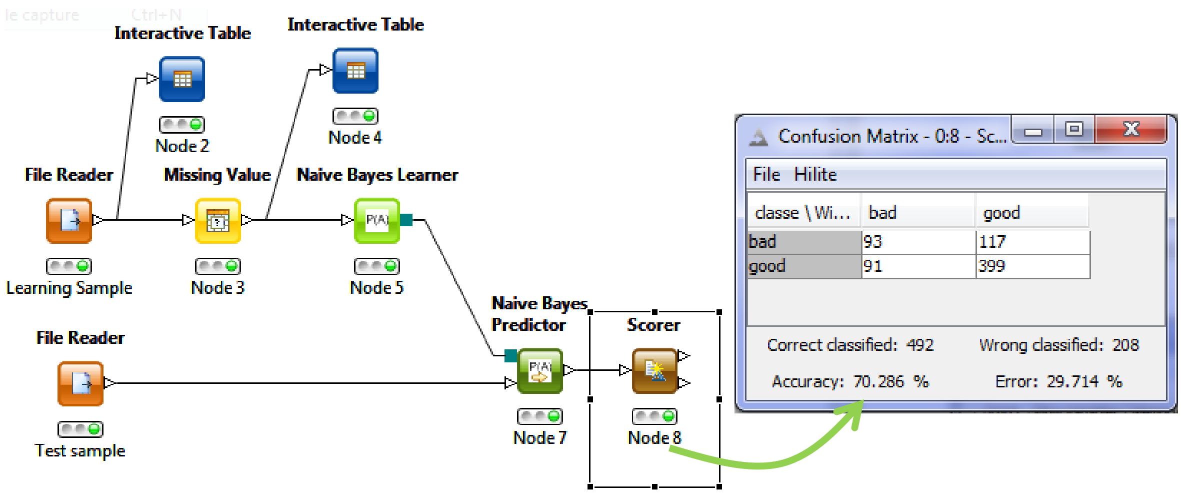
Evaluation on the test set. We want to evaluate the model on the test set. We load this last one by using another FILE READER component. Then, we compute the predicted values using the NAIVE BAYES PREDICTOR tool.



To obtain the confusion matrix, we use the SCORER component. We set (CONFIGURE menu) CLASSE in row and the predicted values [WINNER (NAÏVE BAYES)] in column.



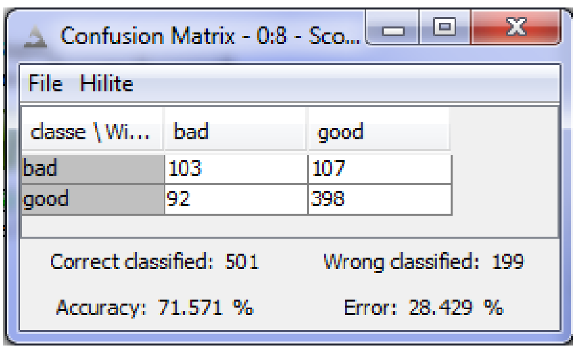
We click on EXECUTE AND OPEN VIEWS. The accuracy rate is **70.286%**.



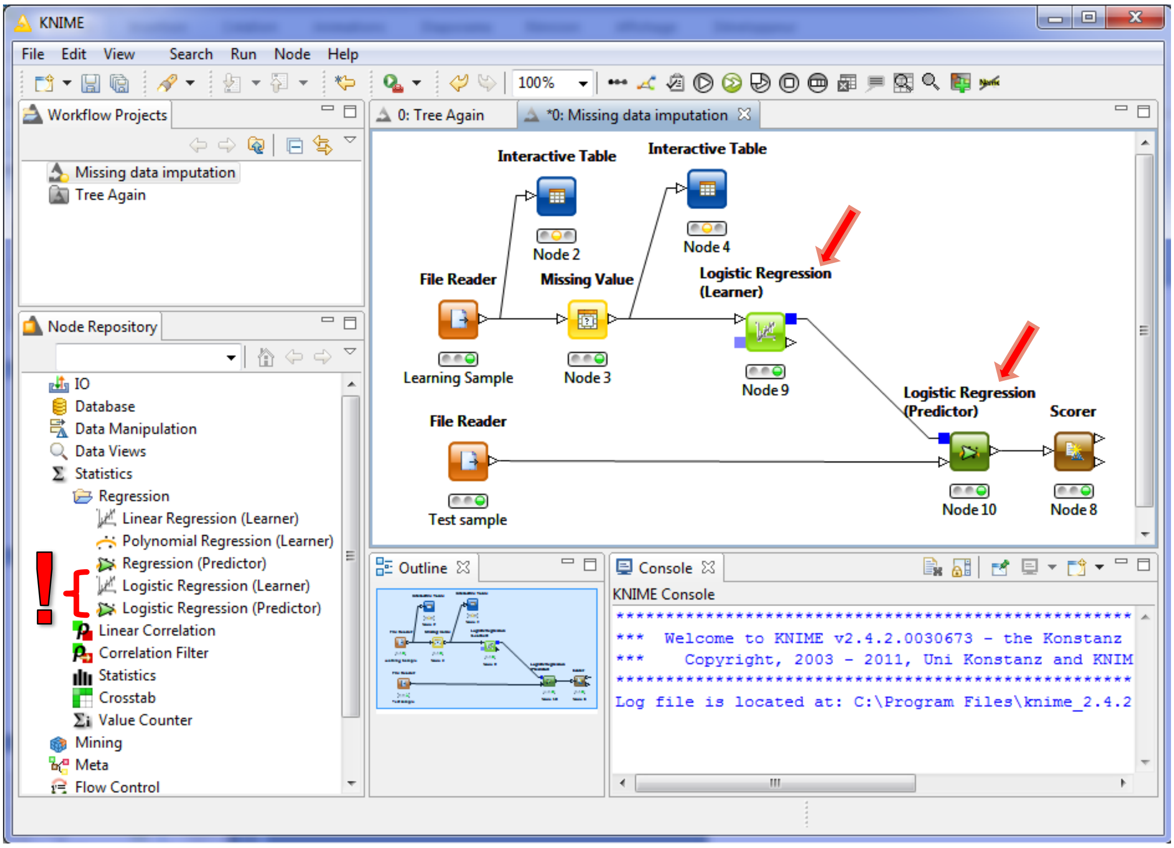
The naive bayes classifier is less penalized by deleting rows than logistic regression. Indeed, the accuracy rate of the model learned from the complete data (without missing values) is 72%. This behavior is probably the consequence of the low number of parameters to estimate from data.

Note: Of course, when we have a high proportion of missing values (e.g. 10%, it remains 40 rows into the learning set), even if the learning process is possible, the model is heavily deteriorated (accuracy rate: 63.429%).

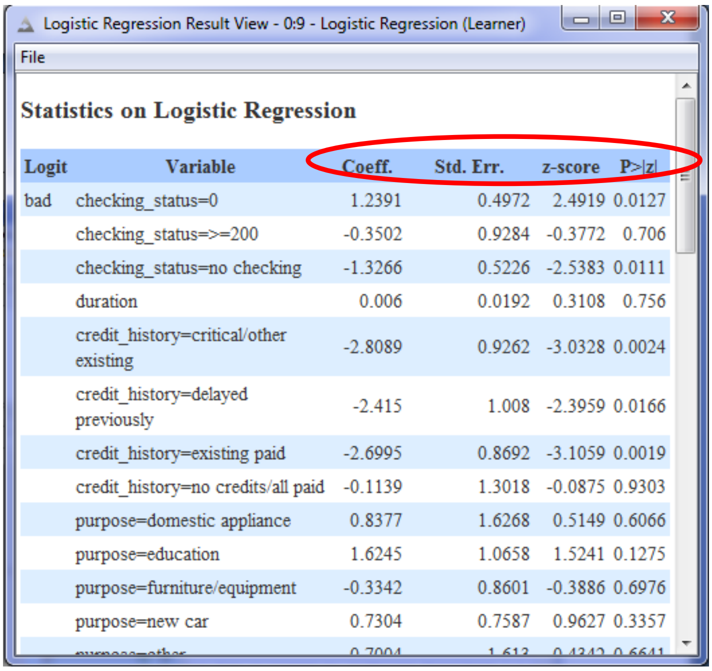
Univariate imputation. When we choose the univariate imputation in the IMPUTE component (mean for integer and double columns, most frequent for the string ones), the accuracy rate becomes 71.571%. For the naive bayes classifier also, as the logistic regression, the univariate imputation is better than the listwise deletion strategy in the MCAR missing values context.



Logistic regression. The LOGISTIC REGRESSION node is available in the STATISTIC / REGRESSION branch into the node repository. I repeat the same analysis using this learning method.



The categorical variables are automatically coded. The output complies to the standards of the domain. We have the estimated parameters, their standard error and test of significance statistic.

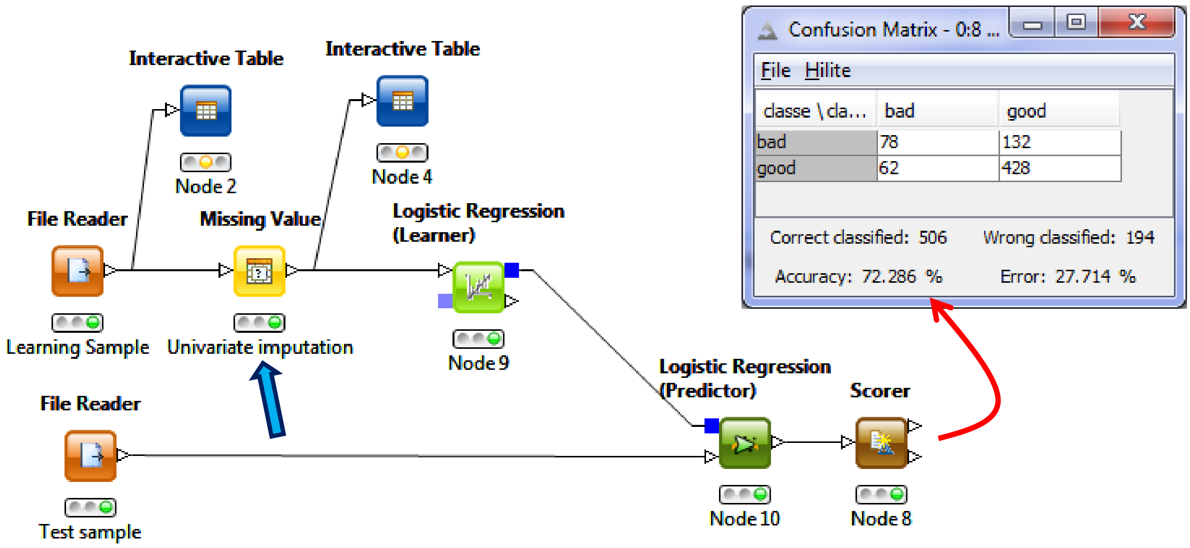

Logistic Regression Result View - 0:9 - Logistic Regression (Learner)

File

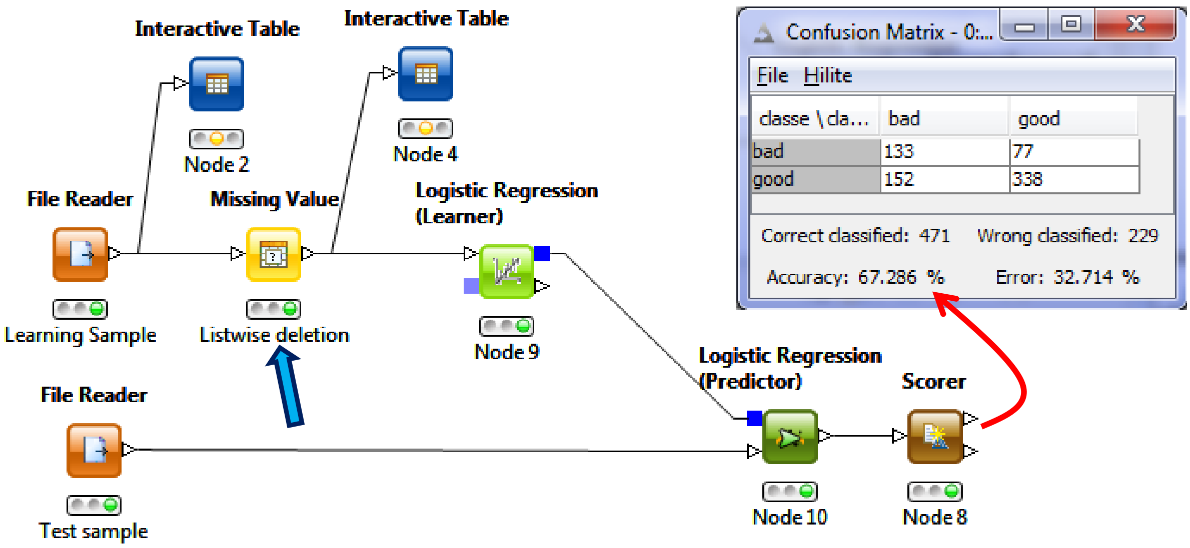
Statistics on Logistic Regression

Logit	Variable	Coeff.	Std. Err.	z-score	P>|z|
bad	checking_status=0	1.2391	0.4972	2.4919	0.0127
	checking_status=>=200	-0.3502	0.9284	-0.3772	0.706
	checking_status=no checking	-1.3266	0.5226	-2.5383	0.0111
	duration	0.006	0.0192	0.3108	0.756
	credit_history=critical/other existing	-2.8089	0.9262	-3.0328	0.0024
	credit_history=delayed previously	-2.415	1.008	-2.3959	0.0166
	credit_history=existing paid	-2.6995	0.8692	-3.1059	0.0019
	credit_history=no credits/all paid	-0.1139	1.3018	-0.0875	0.9303
	purpose=domestic appliance	0.8377	1.6268	0.5149	0.6066
	purpose=education	1.6245	1.0658	1.5241	0.1275
	purpose=furniture/equipment	-0.3342	0.8601	-0.3886	0.6976
	purpose=new car	0.7304	0.7587	0.9627	0.3357
	purpose=other	0.7004	1.613	0.4342	0.6641

We have launched the workflow on the learning set containing 5% of missing values. The test error rate is **72.286%** when we use the univariate imputation.



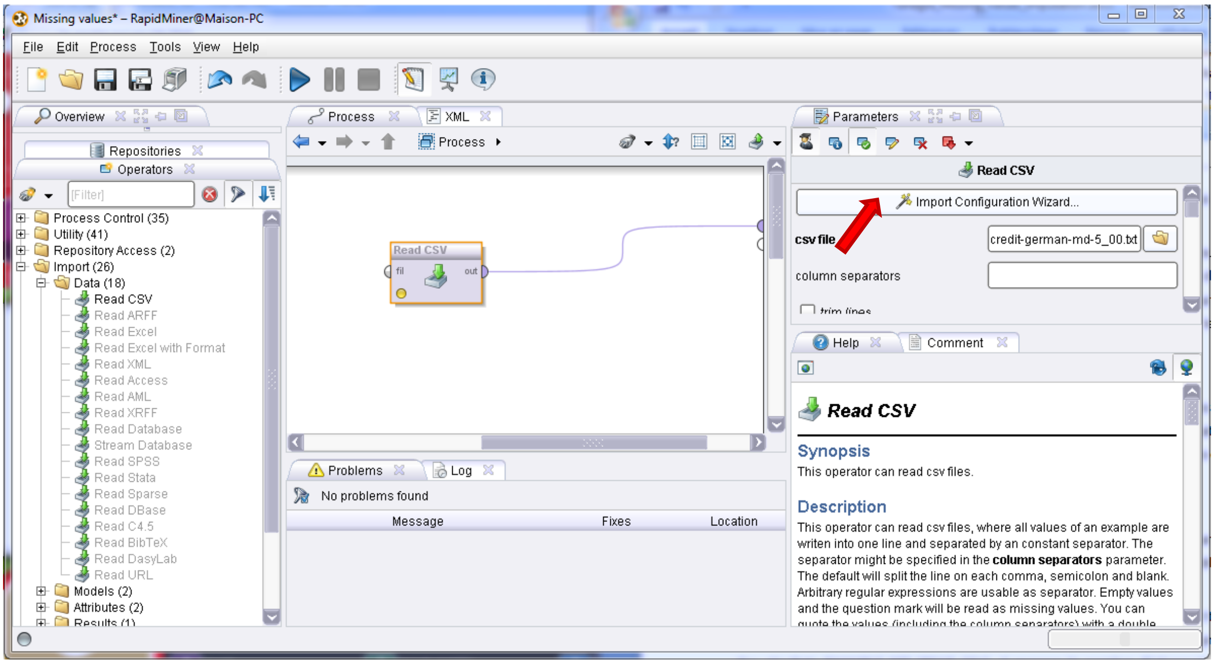
With the listwise deletion strategy, it becomes **67.286%**.



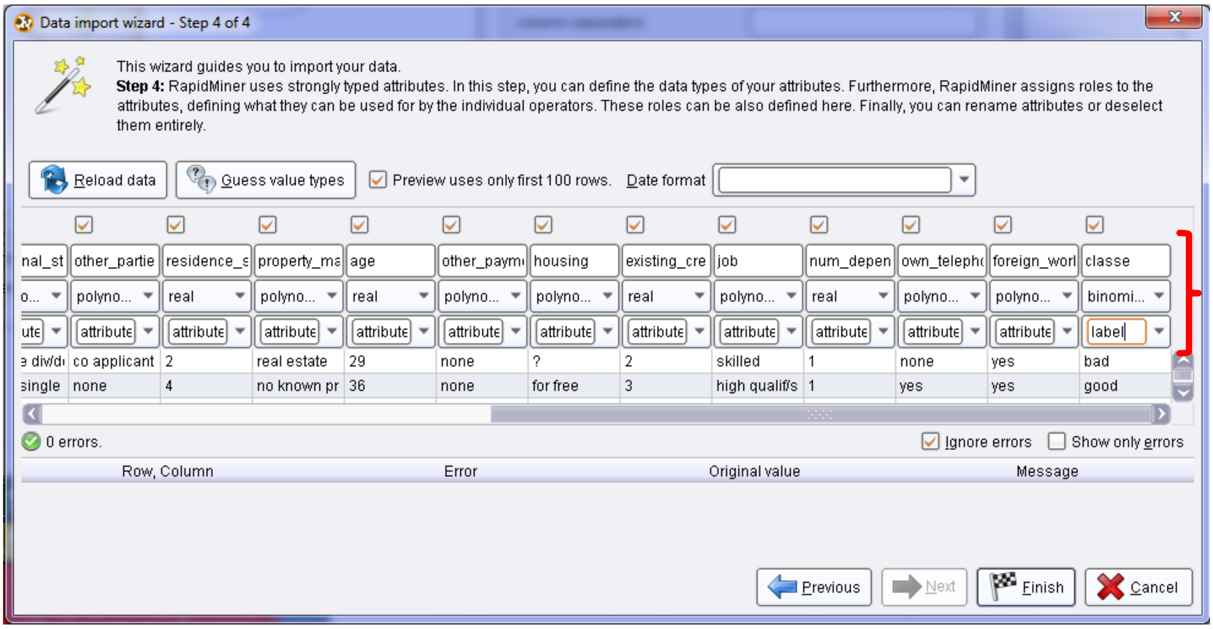
4.3 RapidMiner

Two tools are needed for the detection of missing values in RapidMiner. Thereafter, we use a specific component for their treatment. RapidMiner proposes the listwise deletion and the univariate imputation strategies.

Detection of missing values. After starting RapidMiner, we create a new "Process". We use the READ CSV (IMPORT/DATA) component to import the dataset. The easiest way is to use the wizard (IMPORTATION CONFIGURATION WIZARD button) to set the parameters of the importation.

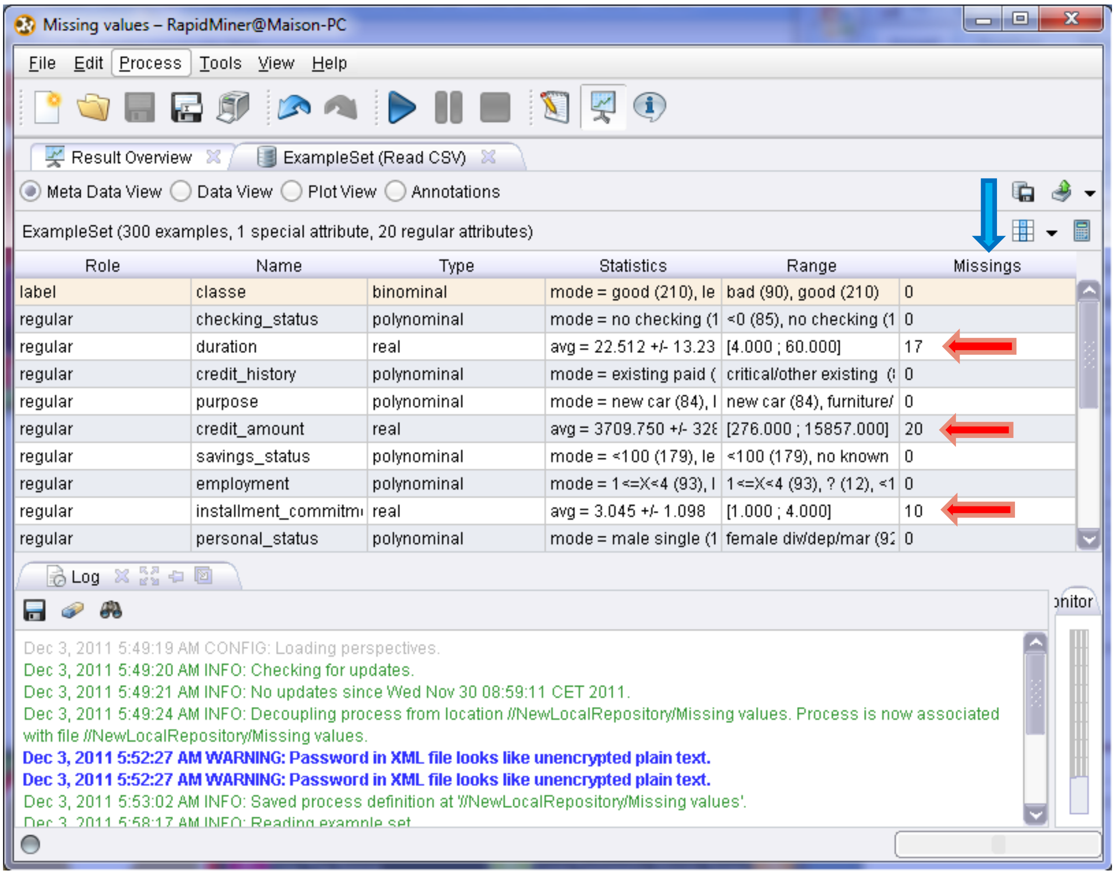


Defining the type of the variable is essential. For our dataset, all the continuous attributes are REAL; the others are POLYNOMIAL (more than two categories) or BINOMIAL (two categories). In addition, we must specify the label column (CLASSE).

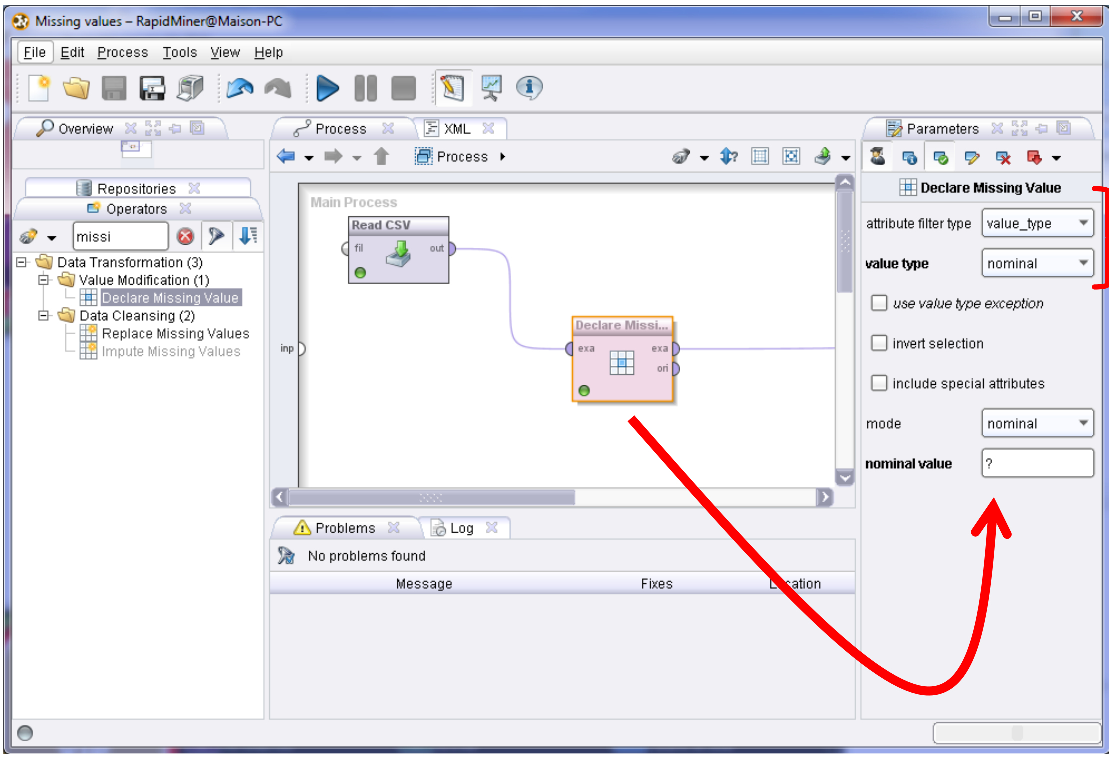


RapidMiner uses this first step to detect missing values. For the REAL columns, the character “?”' is inconsistent with the definition of the variable. The software deducts that there are missing values. We observe this by running the PROCESS (PROCESS / RUN menu).

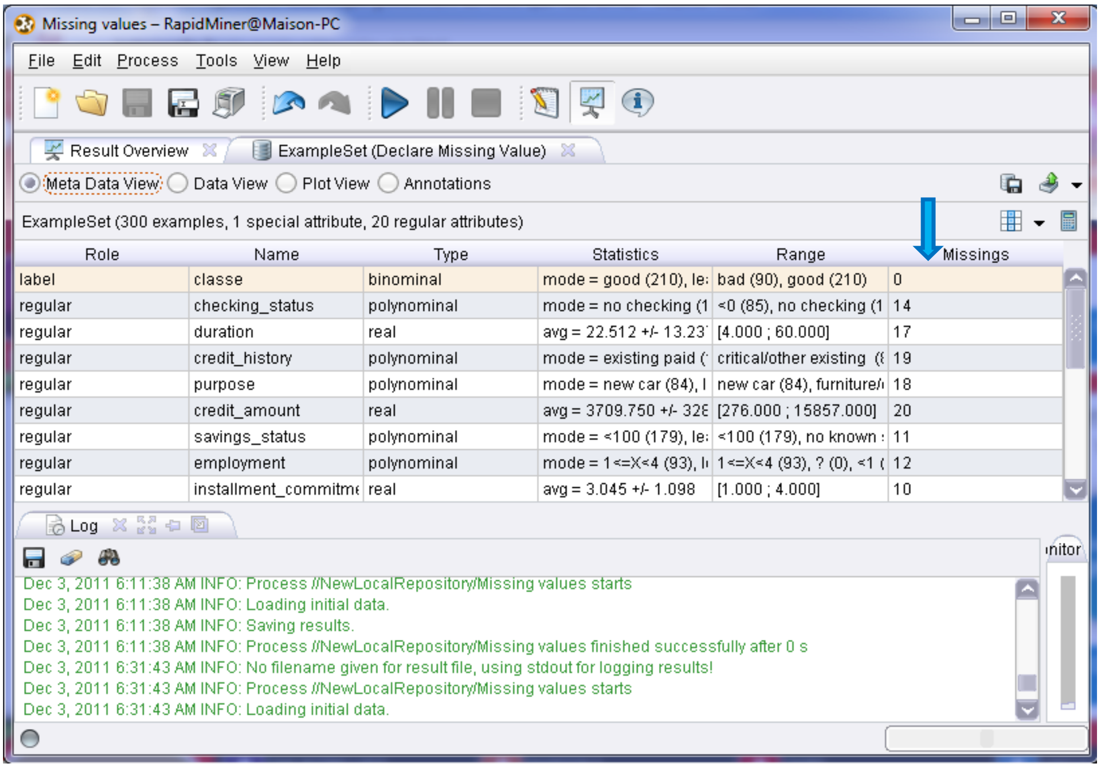
In the EXAMPLE SET (READ CSV) results tab, it lists the number of missing observations for each REAL variable (e.g. 17 for DURATION, 20 for CREDIT_AMOUNT, etc.).



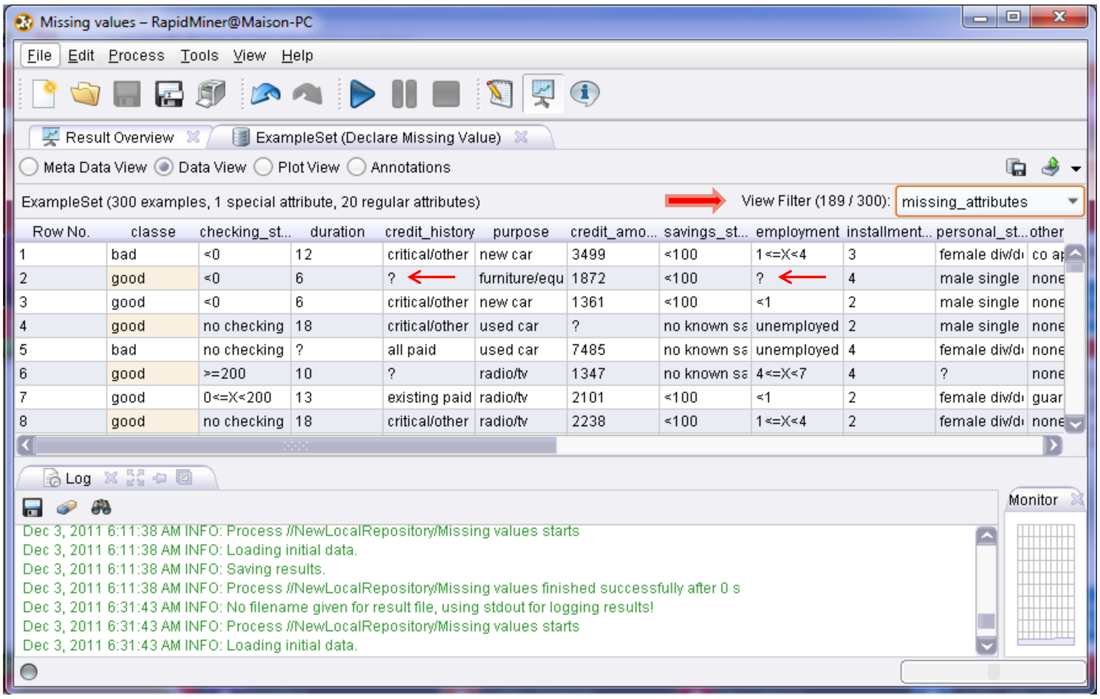
We come back to the PROCESS window. We insert the DECLARE MISSING VALUES component into the workspace. We set that we want to treat only nominal variables (because the detection for the continuous attribute is already completed during the importation process) and we specify the code to missing values ("?").



We obtain the number of missing values for each column[9].



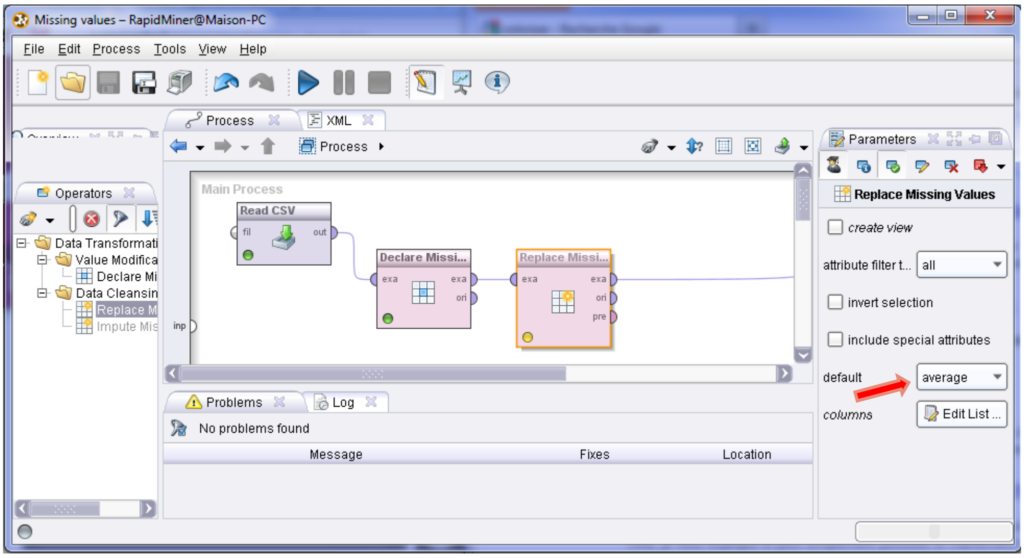
We select the DATA VIEW option into the results tab. We can filter the dataset in different ways. We select for instance the rows with at least one missing value, we obtain 189 rows.



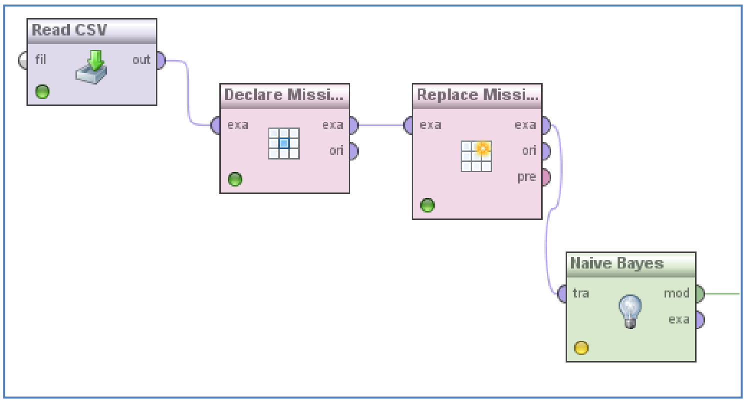
Univariate imputation. We add the REPLACE MISSING VALUES in our process (DATA TRANSFORMATION / DATA CLEANSING). RapidMiner proposes the AVERAGE for the missing values

[9] The whole process seems complicated for a text file such as we want to handle in this tutorial. It becomes easier when we handle a dataset from a DBMS such as Oracle, etc. See http://www.youtube.com/watch?v=0IVZmAk0pI4

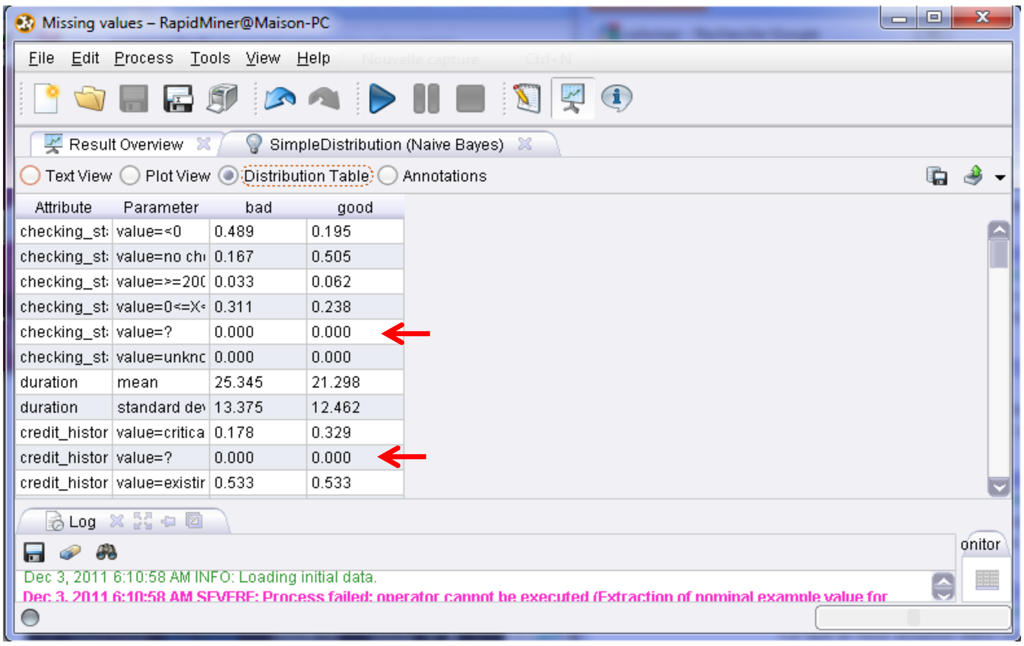
imputation. In reality, this option operates also for the categorical variables. In this situation, RapidMiner uses the MODE.



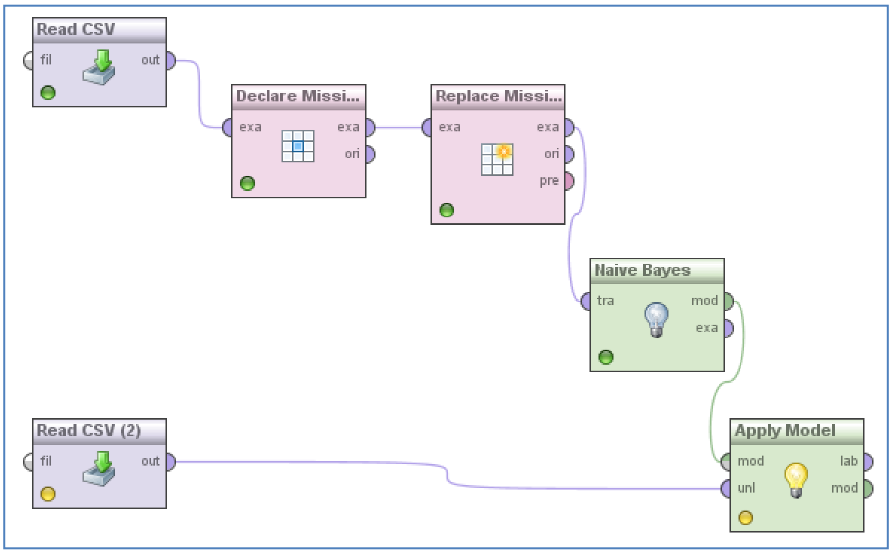
We want to insert the supervised learning component in our process. The logistic regression of RapidMiner does not correspond to the usual method that we find in statistical tools. It seems preferable to use the Naive Bayes classifier here.



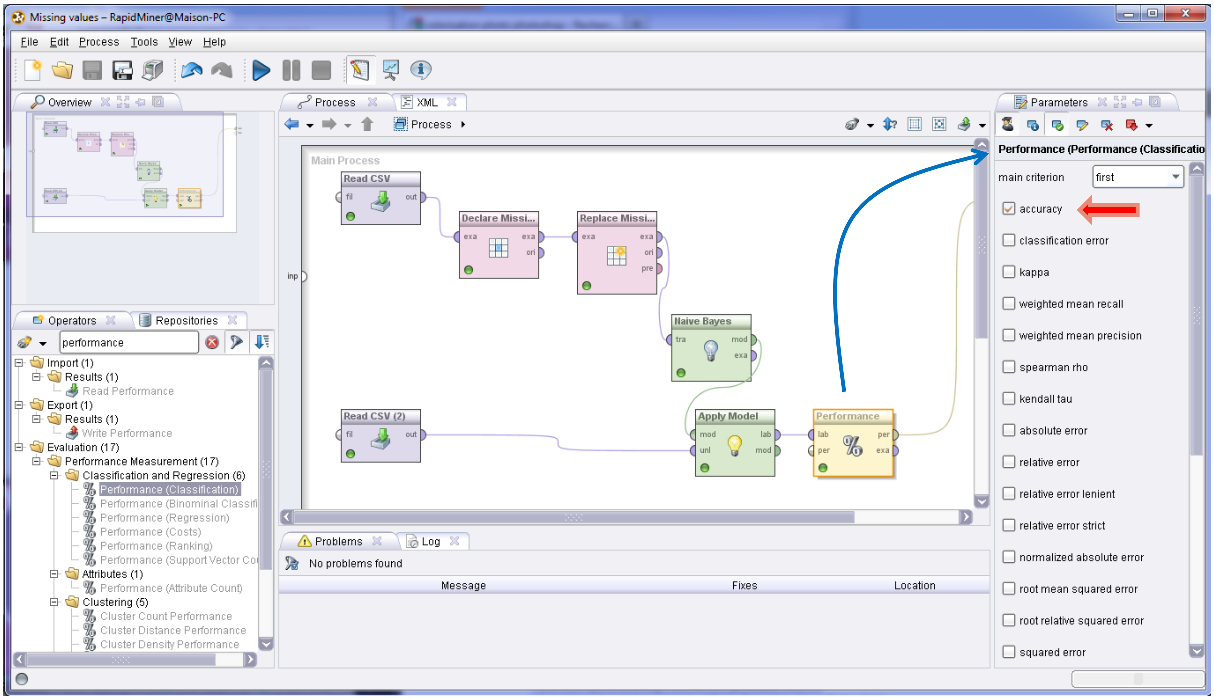
We launch the process. Into the table describing the conditional distributions, we observe that the "?" value is not present. The imputations are really completed.



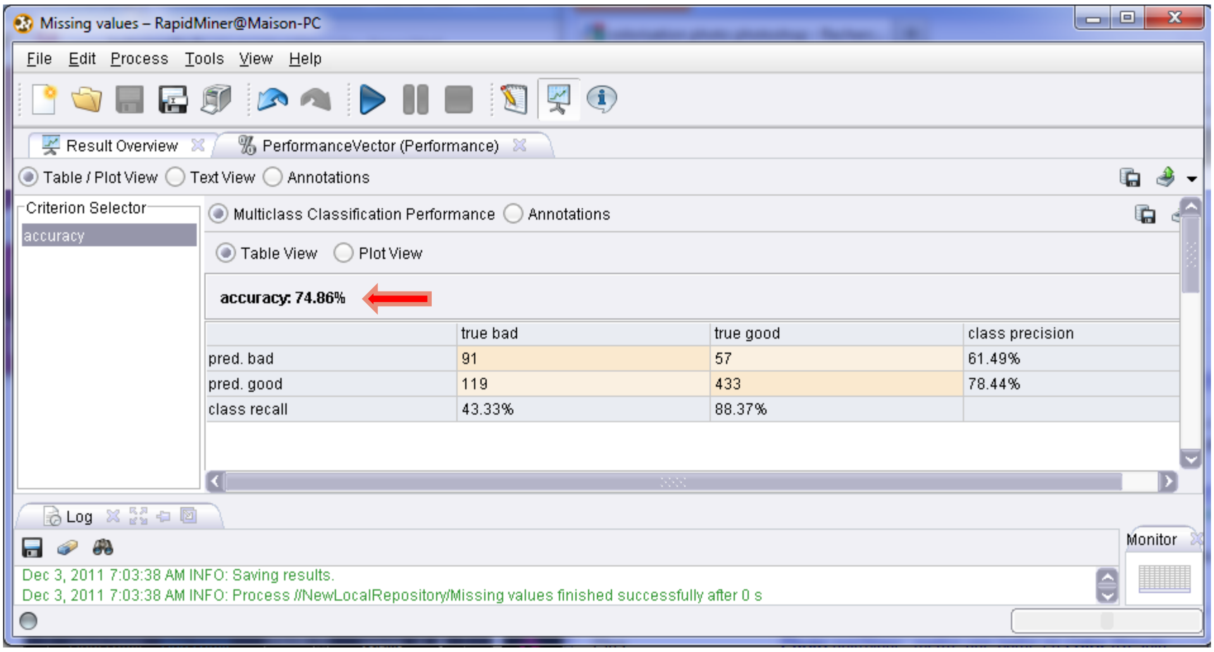
Model evaluation. We insert again the READ CSV component to load the test set (CREDIT-GERMAN-TEST.TXT). We make the following connections between the components.



Last, the PERFORMANCE CLASSIFICATION (EVALUATION / PERFORMANCE MEASUREMENT / CLASSIFICATION AND REGRESSION) component enables to compute the accuracy rate (ACCURACY).



We launch the process. The test accuracy rate is 74.86%.



5 Results on the WAVE dataset (2 classes version)

To confirm the results highlighted in this tutorial, we repeat the same process on the binary version of the waveform[10] dataset (the instances corresponding to the third class are removed). We have 500 instances in the learning set and 32867 instances in the test set. The estimation of the accuracy rate will be more precise.

We obtain the following results using the logistic regression:

WAVE DATASET		Accuracy rate	
% missing	# complete obs.	Listwise Del.	Univ. Imputation
0,00%	500	0,9150	0,9150
0,50%	451	0,9147	0,9150
1,00%	405	0,9082	0,9162
2,00%	331	0,9068	0,9160
5,00%	177	0,8731	0,9174
10,00%	72	0,7847	0,9192
20,00%	6	ERR	0,9188

The results are consistent with those obtained on the GERMAN dataset. When the proportion of missing values increases, the univariate imputation is far better than the listwise deletion strategy.

Curiously, compared with the performance of the model learned from the complete dataset, the accuracy rate of models seems improved when the missing values are replaced by the mean. I have no veritable explication for that. This is probably a consequence of sampling fluctuations. Anyway, the programs and the dataset are available. The reader can repeat the experiments.

[10] http://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+%28Version+1%29

6 Conclusion

The treatment of missing data is a very difficult problem. We must make choices based on factors that we do not control very well (the missingness mechanism). In this tutorial, we have tried to show several software solutions. When the missing values are MCAR, univariate imputation (mean / mode) seems have a good behavior in the logistic regression context, to the extent that our main criterion is the generalization accuracy rate.

7 References

Allison, P.D. (2001), « Missing Data ». Sage University Papers Series on Quantitative Applications in the Social Sciences, 07-136. Thousand Oaks, CA: Sage.

Little, R.J.A., Rubin, D.B. (2002), « Statistical Analysis with Missing Data », 2nd Edition, New York: John Wiley.